Qualitative and Fuzzy Analogue Circuit Design

by

Dipl. (FH) Christoph Reich, M.Sc.

Dissertation

Presented to the Faculty of Computing of

The DeMontfort University at Leicester

in Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The DeMontfort University at Leicester England

February 18, 1999

Abstract

Predicting and reasoning about the behaviour of physical systems is a difficult and important problem. It is essential for many complex engineering' tasks such as designing, monitoring, controlling and diagnosis. The process of design is one of the most challenging tasks for engineers since it involves modelling a system that does not yet exist and deriving its behaviour by simulation. In this situation neither the conditions of the complex and non-linear relationships which model physical systems nor the models themselves are completely and precisely known. It is only possible to define a model whose parameters and/or initial conditions are known imprecisely.

Analogue electronic circuits are excellent examples of systems that have been used to show the need to handle imprecision. There are many ways for handling imprecision including qualitative reasoning and fuzzy logic.

This thesis is concerned with the specification, design, and simulation of imprecise non-linear systems using a qualitative fuzzy approach. There are three main outcomes presented in the thesis:

First, the representation of imprecise analogue signals and models of systems known imprecisely has been developed using Fuzzy Relation Memories. Specifying imprecise signals by Fuzzy Relation Memories enable hierarchical structures by combining them using ordinary arithmetic operations. Modelling imprecise non-linear system by Fuzzy Relation Memories extends the known types of fuzzy systems.

Second, a new approach, called the 'Interactive Evolutionary Algorithm Approach', for solving imprecise differential equations has been developed. This ap-

proach solves ordinary differential equations that have imprecise differential equation parameters and/or imprecise initial values.

Third, using the Fuzzy Relation Memories and the Interactive Evolutionary Algorithm, a high-level framework for Imprecise Analogue Circuit Design (IACD) has been developed. This framework is specifically designed to support the design engineer at the circuit paper prototype level.

Acknowledgments

I would like to thank my advisors, Christian Horn, Tom Routen and Peter Innocent, for their support throughout my PhD work. They helped me keep my standards high and provided me with the opportunity to explore lots of interesting paths. Especially thank to my local supervisor Christian Horn and my supervisor Peter Innocent at the DeMontfort University, who helped, guided, and supported me during the time of domain knowledge acquisition, how to do research, writing papers, and focusing on important parts of the PhD work.

Contents

List of Definitions

List of Figures

Chapter 1

Introduction

Predicting and reasoning about the behaviour of physical systems is a difficult and important problem. It is essential for many complex engineering' tasks such as designing, monitoring, controlling, or diagnosis. The knowledge about physical systems in the engineering world is often incomplete and imprecise. Neither the conditions of the complex and nonlinear relationships which model physical systems nor the models themselves are completely and precisely known. To demonstrate the necessity for handling incompleteness and imprecision the process of analogue circuit design has been chosen.

1.1 Design of Analogue Circuits

Designing analogue electronic circuits has been used as an example to show the necessity to handle incompleteness and imprecision throughout this thesis. The process of design is one of the most challenging tasks for electronic engineers. It involves common sense and expert knowledge which is always imprecise and incomplete. No one understands down to the last detail how any mechanism actually works. Even if it were possible to construct a model of, say, a television set, down to the level of quantum electrodynamics, the model would be absurdly, uselessly large. At the

beginning of the design process a very vague model of a possible real system is created with the incomplete and imprecise knowledge available at the current state of the design process. A model of the wanted system and the specifications are refined, using imprecise simulation, until a real system matches the model and the specifications, the design process is finished.

The main concern of this thesis is to support designers by the design task at a very high-level. At this level of design imprecision and incompleteness is involved in modelling and specifying of analogue circuits. VHDL-A 1076.1 [VHDL-A Standard 99, 1999] for specifying and simulating continuous systems or the most spread simulator SPICE [Nagel, 1973] do not take into account imprecision or incompleteness. For the analysis of imprecise defined models a new simulation approach, the **qualitative fuzzy simulation**, has been developed.

1.2 Goals of this Research

Main goal of this thesis:

This dissertation is concerned with the specification, design, and simulation of imprecise nonlinear systems using a qualitative fuzzy approach.

This thesis is divided into two main parts:

- The first part of the thesis discusses the qualitative and fuzzy approach, the representation of imprecise knowledge, and the simulation of imprecise defined systems. The thesis presents the general foundation for the development of a design tool which can handle systems not known exactly.

- The second part is an application prototype description in the domain of analogue circuit design. It describes a framework for the design of analogue circuits that can handle imprecise knowledge involved in the specification, modelling and simulation of analogue electronic circuit design process.

1.3 Summary of Results

There are three main results presented in this research:

1. The representation of imprecise analogue signals and modelling of systems not known precisely by **Fuzzy Relation Memories (FRMs)** that allow further processing by arithmetic operations (Chapter 4).

2. The **Interactive Evolutionary Approach** for solving ordinary differential equations that have imprecise defined differential equation parameters and imprecise initial values (Chapter 6).

3. The high-level framework for **Imprecise Analogue Circuit Design (IACD)** has been developed that uses the new findings of 1. and 2. (Chapter 9).

1.4 Reader's Guide

This thesis consist of two parts subdivided into 13 chapters completed by 4 appendixes. Here is what they contain:

Chapter 1 (this chapter) gives an introduction and an overview of this work.

Part I *Qualitative Fuzzy Simulation*: Builds the theoretical foundation of the thesis.

Chapter 2 *Historical Survey of Qualitative Fuzzy Simulation*: A review of the qualitative approach and the fuzzy approach is given with advantages and disadvantages of the approaches. A discussion of historical attempts to combine qualitative simulation and fuzzy simulation to qualitative fuzzy simulation is given.

Chapter 3 *Representation of Imprecise Values*: Discusses and defines the representation of imprecise numbers, imprecise intervals, linguistic variables, linguistic hedges, qualitative values, qualitative numbers, qualitative intervals, that are the basis for specifying imprecise systems.

Chapter 4 *Representation of Imprecise Analogue Signals*: **Fuzzy Relation Memories (FRMs)** — **Fuzzy Curves** are derived from fuzzy relation, fuzzifying functions, and temporal fuzzifying functions. FRMs are used for modelling imprecise nonlinear systems and for quantities varying during simulation which are called imprecise signals in this thesis.

Chapter 5 *Constraints — Relations*: For combining imprecise signals or build up hierarchical structures of imprecise models, in this chapter the algebraic operations, equality, similarity, addition, subtraction, multiplication, division, derivation, and integration of **Fuzzy Relation Memories**, supported by many examples are defined.

Chapter 6 *Simulation of Imprecise Ordinary Differential Equations*: Existing approaches for solving differential equations not known exactly are discussed. More, a new approach is introduced: **Interacting Evolutionary Algorithm Approach**. A 1st-order and a 2nd-order differential equation example illustrates the successful simulation of the approach.

Chapter 7 *Imprecise Modelling*: Four kinds of modelling: functional based, rule based, equation based and the new **Fuzzy Relation Memory** modelling is considered and illustrated by examples.

Chapter 8 *Qualitative Fuzzy Simulation*: Starts with a general discussion of simulation, defining *Approximate Model-Based Reasoning*, and summarizes the features of the new *qualitative fuzzy approach.*

Part II *Qualitative and Fuzzy Analogue Circuit Design*: This part uses the methods and approaches introduced in Part I on the problem of analogue circuit design.

Chapter 9 *High-Level Framework for Imprecise Analogue Circuit Design (IACD)*: From a general definition of design a specific definition for engineering design has been developed. The first idea of a circuit which most likely meets the specification is drawn on paper by the design engineer. Designing circuits on a piece of paper is called the **circuit paper prototype** design by this thesis. At this level of abstraction the findings of Part I are most important for modelling and simulation. Including the circuit paper prototype a high-level framework is developed and described. This high-level framework consists of three major design steps. First the circuit paper prototype design, second the qualitative configuration design, and third the fuzzy configuration design.

Chapter 10 *Defining Imprecise Specifications*: It outlines the specifications of nonlinear systems not known exactly. It describes the user interface for specifying imprecise values. Especially the specification and the use of **Fuzzy Relation Memory** is discussed in this chapter.

Chapter 11 *Circuit Paper Prototype Design*: Examines the implementation of circuit paper prototype design in detail. It points out the different modelling aspects interesting for the designer and shows an example how at this level of design a designer can be supported.

Chapter 12 *Qualitative and Fuzzy Configuration-Design*: Demonstrates the configuration design at the qualitative and fuzzy level in detail. For configuration-design a pre-simulated circuit cell database is needed. It is showed, how such a database would look.

Chapter 13 *Conclusions*: Completes this work by a discussion of the previous chapters and drawing a conclusion.

Appendix A *Fuzzy Reasoning Basics*: Definitions of non interactive, strongly positive interactive, and, strongly negative interactive addition, subtraction, mul-

tiplication, and, division of Triangular Fuzzy Numbers and Triangular Fuzzy Intervals

Appendix B *Historical Survey of Analogue Circuit Design Tools*: Discusses the present status of existing analogue circuit design tools and frameworks. They are categorized in bottom-up and top-down approaches. The need for imprecise specification, imprecise reasoning, during the design of nonlinear systems, and simulation is motivated.

Appendix C *Pre-Simulated Database*: Displays a number of circuit cells pre-simulated using SPICE [Nagel, 1975]. The circuit cells are listed with their SPICE-file, black-box description, input data, and output data.

Appendix D *Software of the CD*: Gives an overview of the contents of the CD.

Part I

Qualitative Fuzzy Simulation

"As the complexity of a system increases, our ability to make precise and yet significant statements about its behaviour diminishes until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics." by Lotfi Zadeh [Zadeh, 1973].

The real world is complex; complexity in the world generally arises from uncertainty in the form of ambiguity. Problems featuring complexity and ambiguity have been addressed subconsciously by humans since they could think. Humans are able to reason approximately, a capability that computers currently do not have. A step towards improving computers are approaches based on qualitative and fuzzy techniques. The approach of this thesis is a combination of the qualitative and the fuzzy approach, called **Qualitative Fuzzy Simulation**.

1.5 Overview of Part I

This part of the dissertation is the theoretical foundation for **Part II** which uses the findings of this part in the domain of analogue circuit design. Naturally, design is a very vague process that is taken into account in the qualitative fuzzy approach. **Part I** starts with a review of qualitative and fuzzy simulation (Chapter 2). It discusses some existing approaches that combine qualitative simulation and fuzzy simulation to qualitative fuzzy simulation. It also describes in detail the qualitative fuzzy simulation approach is given in Chapter 8. It answers the three most essential questions when simulating a system. They are:

1. *How are the model parameters represented?* Various imprecise value representations are given in Chapter 3. In Chapter 4 of the thesis the representation of imprecise analogue signals or modelling of imprecise nonlinear systems by **F**uzzy **R**elational **M**emories (FRMs) is emphasized.

2. *How does the simulation work?* FRMs are added, subtracted, multiplied, divided, derivated, or integrated shown in Chapter 5. Solving imprecise ordinary differential equations is done by a new method; the **Interactive Evolutionary Algorithm Approach**; introduced in Chapter 6.

3. *What possibilities are available for modelling?* There are several kind of modelling possibilities described in Chapter 7. Most interesting is the new kind of modelling using **Fuzzy Relational Memories**.

Chapter 2

Historical Survey of Qualitative Fuzzy Simulation

This chapter provides background information about qualitative and fuzzy reasoning about physical systems. After listing advantages and disadvantages of the qualitative approach a historical survey of fuzzy simulation is given, tabulating the benefits and drawbacks. The recently appeared approaches that combine qualitative simulation and fuzzy simulation are discussed and summarized.

2.1 Historical Survey of Qualitative Simulation

Humans have certainly been reasoning *qualitatively* about the physical world as long as they have been able to reason at all. Even though systems of measurement have been known for a long time, people are typically unaware of the precise measures of the things around them. People do, however, have a very keen ability to detect distinctions between things. Reasoning about the distinctions between properties that have precise, but unknown, values is qualitative reasoning. Qualitative simulation is used for predicting the behaviour of qualitative models. Werthner [Werthner, 1994] wrote:

> "Qualitative reasoning deals with the modelling of dynamic systems and describes their structure and behavioural changes in time." [Werthner, 1994].

From this qualitative reasoning and qualitative simulation is defined as:

Definition 2.1 (*Qualitative Reasoning*): Qualitative reasoning is searching for qualitative models and predicting their behaviours by qualitative simulation.

Definition 2.2 (*Qualitative Simulation*): Qualitative simulation seeks to automate the human ability to predict behaviours about the physical world.

Qualitative reasoning, qualitative simulation, naive physics, and qualitative physics are synonymously used in the literature. The original focus was on the common sense reasoning that underlies everyday life (e.g. parking cars). Nowadays it is reasoning about physical systems in the domain of experts. Farquhar [Farquhar, 1993] stated two important goals of qualitative reasoning still valid today:

1. It provides the strongest inference possible, given incomplete information, and
2. it requires only the distinctions necessary to answer a query.

2.1.1 Historical Milestones of Qualitative Reasoning

The most wide-spreading works about qualitative reasoning were published in the *Journal of Artificial Intelligence* 1984 (Vol. 24) and 1993 (Vol. 59). Three major approaches were significant for most of the work done in this area. Their principle differences are based on the ontological primitives used to describe the physical system and the mechanisms to interpret the behaviour determined from the qualitative reasoning over physical quantities. These three historic milestones of the qualitative reasoning are:

- **ENVISION by De Kleer and Brown [de Kleer and Brown, 1984].** ENVISION uses components as primitives which are connected to build a system. The behaviour of the overall system is derived from the behavioural interaction between the components through their connections. The components are the causal reason for the total behaviour of the system, typically called device-centered approach. Further developments based on the device-centered approach were [de Kleer and Brown, 1984], [de Kleer, 1984], [de Kleer and Brown, 1986], [Iwasaki and Simon, 1986], and [Williams, 1984].

- **QPT (Qualitative Process Theory) by Forbus [Forbus, 1984].** This process-oriented approach defined processes as the basic primitives. Processes influence and change objects (i.e. the values of their variables). These variables describe the attributes of individuals which participate in a specific situation. The total behaviour of the system is determined by the interaction of processes. In QPT the components are passive objects which can only be changed by processes. Publications dealing with this approach are from [Forbus, 1984],[Forbus, 1993], and [Martschew, 1988].

- **QSIM (Qualitative SIMulation) by Kuipers [Kuipers, 1984].** This constraint-based approach contains no ontology for the physical situation. QSIM describes devices or situations directly by a set of variables and qualitative differential equations (QDEs) or relations. Besides [Kuipers, 1984], [Kuipers, 1986], [Kuipers, 1993b], [Kuipers, 1993a], [Kuipers, 1994], and [Kuipers and Åström, 1994] there are a huge number of publications and extensions using QSIM partially discussed in subsection 2.1.3.

2.1.2 Advantages and Disadvantages of Qualitative Reasoning

Qualitative simulation offers several advantages over conventional numerical simulation of mathematical models:

- **Handling incomplete information.** Very often there is insufficient quantitative information available to perform accurate simulation, e.g. system design. The technology of qualitative modelling and qualitative simulation makes it possible to build a simulation model and perform prediction even with incomplete knowledge of the system.

- **Imprecise but correct complete prediction.** Qualitative simulation can proceed with simulation even when a precise numerical model is not available. Given an incomplete specification all possible courses of behaviour will be generated by qualitative simulation. To perform a numerical simulation, we must assume parameters, even if values are not known. On such assumptions depend the validity of the numerical simulation which is pretending a correct and precise accuracy of the prediction. It is more desirable to see an accurate but less precise prediction than a very precise one that might be incorrect.

- **Easy exploration of alternatives.** Usually the designer must perform many conventional numerical simulations to see the entire range of possible behaviours. Because one precise simulation produces only one behaviour of all possible behaviours. Qualitative simulation can produce in one run all possible behaviours. Given incomplete specification of a system the qualitative simulation produces the entire range of possible behaviours at once. Qualitative simulation gives the designer an overview of a large space of possibilities.

- **Automatic interpretation.** Numerical simulation produces detailed numerical output that describe the behaviour over time. To make sense of the output,

a domain expert, knowledgeable about the phenomena being simulated, must interpret the numbers to identify important qualitative characteristics of the behaviour. Qualitative simulation produces predictions directly in terms of important qualitative characteristics of the behaviour. The result of a qualitative simulation requires much less effort to interpret.

Disadvantages of the early qualitative reasoning approaches:

- **Qualitative reasoning generates ambiguous behaviour.** Qualitative simulation can predict a number of qualitative behaviours, that are difficult to survey. Some of the predicted behaviour is *spurious behaviour* which can not be seen on any concrete real system. Qualitative models may represent an entire class of numerical models, the different solutions correspond to these models substantially enlarge the solution set.

- **Qualitative reasoning is computationally expensive.** Qualitative simulation does not always produce a definite behaviour but a tree of possible solutions, called *envisioning tree*. Simulation of complex dynamic systems can produce an envisioning tree exponentially growing with the number of variables when applied to complex systems [Milne, 1991]. The reasons for the exponentially growing simulation space are:

  - Landmarks can be pure symbolic variables, no real number must be known.
  - The qualitative domain is defined over the complete real number space. The intervals between landmarks are considered, too.
  - Qualitative values can only be ordered in time.

- **Qualitative arithmetic is not associative.** Besides the qualitative sign arithmetic[1], the arithmetic using a finite set of qualitative quantities is not associative.

[1]The qualitative domain is defined as:

- **Qualitative reasoning has limited time modelling.** The simulation time involved in dynamic systems simulation are qualitative values, too. The only possibility for ordering the states is the instrument of partial ordering. Which simulation state of two reaches a particular time point first is difficult to determine and called the ordering problem. This results in a huge set of possible behaviours. It is not possible to propagate the duration time of a state or the effect of an influence over a longer time.

Definition 2.3 (*Ordering Problem*): Two or more parameter are reaching one and the same landmark. In general it is unpredictable which of the parameters reach the landmark first, described as the *ordering problem.*

The consequence of the ordering problem is that different behaviours are generated.

2.1.3 Recent Works of Qualitative Reasoning

Recently there have been different approaches to overcome some of the limitations of the known qualitative reasoning technologies, like:

- **Combining mathematical methods and qualitative technologies.** Qualitative reasoning is used to automize typical mathematical techniques and interpret their results. The system developed by [Yip, 1991] analyzes non-linear dynamic systems by doing numerical experiments and displaying the graphs in a phase diagram. It reasons from specific qualitative characteristics of such graphs and decides which numerical experiment is sensible to proceed.

- **Order of magnitude reasoning.** Humans commonly use qualitative reasoning techniques which use information about relative orders of magnitude of

$$[\,] : R \rightarrow Q_R, [x] := \begin{cases} \boxed{+} & if\ x > 0 \\ \boxed{O} & if\ x = 0 \\ \boxed{-} & if\ x < 0 \end{cases}$$

quantities describing an analyzed system, as stated in [Raiman, 1991]. Order of magnitude reasoning can reduce the burden of calculations in qualitative simulation.

- **Temporal reasoning.** The Q3 system of [Berleant and Kuipers, 1998] and FUSIM [Shen and Leitch, 1993] use time intervals and derivations to solve the partial ordering problem. Davis shows how information about order of magnitude of rising of different parameters are used to predict time ranges [Davis, 1989]. A constraint system by Williams [Williams, 1986] which labels values by time intervals (episodes) introduces temporal reasoning to qualitative simulation.

- **Inclusion of quantitative Information.** In many situations there exists quantitative numerical information which can be used to eliminate spurious behaviour (in reality non-existing qualitative behaviour). In the system monitoring work of [Forbus, 1987] and [DeCoste, 1991] the quantitative measurements of a system are transfered to qualitative measurements and the predictions of the qualitative simulation are tuned with the quantitative measurements of the system. The system Q3 [Berleant and Kuipers, 1992] is an extension of QSIM to include quantitative information. The qualitative landmarks are restricted by the use of intervals of possible quantities. Q3 uses quantitative information to eliminate some of the possible *spurious behaviour*.

2.2 Historical Survey of Fuzzy Simulation

Fuzzy simulation is a sub-term of the more common known term fuzzy logic which was well stated by McNeill and Freiberger [Mc Neill and Freiberger, 1994] in "Fuzzy logic — The Revolutionary Computer Technology that is Changing Our World"::

> "Fuzzy logic is a charming name and a wild misnomer. Fuzzy logic is not logic that is fuzzy, but logic that describes and tames fuzziness. And

> even that definition falters, for most of the theory is not logic at all. It is a theory of fuzzy sets, sets that calibrate vagueness."

It all began in 1964, the year Lotfi Zadeh invented and christened fuzzy logic. In his 1965 paper: ""Fuzzy Sets" ([Zadeh, 1965]), Zadeh lay the foundations of the fuzzy logic today and explained its importance. The huge amount of published papers and very often cited person shows that no one else influenced fuzzy logic over so many years than Lotfi Zadeh. The book: "Fuzzy Sets and Applications: Selected Papers by Lotfi A. Zadeh" edited by Ronald R. Yager [Yager et al., 1987] gives a good overview of the work done by Lotfi Zadeh.

A primary force in advancing the practical implementation of fuzzy theory has been Japanese researchers; they commercialised this technology and have now over 2000 patents in this area. Most impact appeared in the area of fuzzy control systems and represent the largest part of all commercial applications (see [Sugeno, 1985]). The Sendai subway [Yasunobu and Miyamoto, 1985] is the prototypical example application of fuzzy control system. A fuzzy control system simply described has the purpose to influence the behaviour of a system by changing an input or input to that system according to a rule or set of rules that model how the system operates. In fuzzy control systems there have been countless publications and applications. The Fuzzy Logic notation is a generic term summarizing fuzzy sets, fuzzy relations, fuzzy rule-based systems, fuzzy simulation, fuzzy reasoning, fuzzy decision, fuzzy classification, fuzzy pattern recognition, fuzzy control systems, approximate reasoning, etc.

2.2.1 Fuzzy Reasoning and Fuzzy Simulation

The emphasis of this thesis is founded in the subject of approximate reasoning about imprecise[2] defined nonlinear systems. Generally approximate reasoning was

[2]One note of clarification: in exact terms imprecision is different from uncertainty. Uncertainty, as in probabilistic or fuzzy inference, refers to the degree to which we believe a certain fact.

described by Lotfi Zadeh ([Zadeh, 1979]) as:

> "Informally, by fuzzy reasoning or, equivalently, approximate reasoning we mean the process or processes by which a possibly imprecise conclusion is deduced from a collection of imprecise premises. Such reasoning is, for the most part, qualitative rather than quantitative in nature, and almost all of it falls outside of the domain of applicability of classical logic." by Lotfi Zadeh ([Zadeh, 1979]).

Fuzzy reasoning about physical systems is defined as follows:

Definition 2.4 (*Fuzzy Reasoning*): Fuzzy reasoning deals with the modelling of physical systems known imprecisely and describes their behaviour using these models.

These systems cannot be simulated with conventional crisp or algorithmic approaches but they can be simulated because of the presence of other information — observed or linguistic — using fuzzy simulation methods. The concept of linguistic variables and its application to approximate reasoning is illustrated from different angles by Zadeh's papers [Zadeh, 1973], [Zadeh, 1975a], [Zadeh, 1975b], [Zadeh, 1975c], and [Zadeh, 1979].

Definition 2.5 (*Fuzzy Simulation*): Fuzzy simulation is predicting the behaviour of models known imprecisely, which are fuzzy models, based on the fuzzy logic theory. Fuzzy nonlinear simulation is predicting the behaviour of imprecise nonlinear models.

Precision refers to the precision of the factual statement itself. For example: In the boiling-water example, a given fact might be that the water's initial temperature is somewhere between freezing and boiling. This is an imprecise statement because it does not say exactly what the temperature is. However, it is not an uncertain statement, because the statement is 100% true (or believed to be true). An uncertain statement in this case would be "it is 90% certain that the temperature is 30°C. In this thesis the term uncertainty is used in form of ambiguity and therefore uncertainty is synonymously used to imprecision.

One suitable representation of simple and complex systems is by fuzzy rule-based systems. A fuzzy rule-based system consists of

1. a set of rules that represent the engineer's understanding of the behaviour of the real system,

2. a set of input data observed going into the real system, and

3. a set of output data coming from the real system.

The input and output data can be numerical, or they can be non numeric observations in form of linguistic statements (e.g. low, big, tall, etc.). A general fuzzy model for nonlinear simulation is described by Kosko's paper: "Fuzzy Systems as Universal Approximators" [Kosko, 1992]. The model of a system consists of IF-THEN rules.

A fuzzy system is a model of the real system. It contains a set of fuzzy rules stated by experts or algorithms, e.g. neural nets or statistical procedures. Different experts and algorithms generate different sets of fuzzy rules and so approximate differently the real system. These IF-THEN rules map input fuzzy sets to output sets, shown in Fig.2.1.

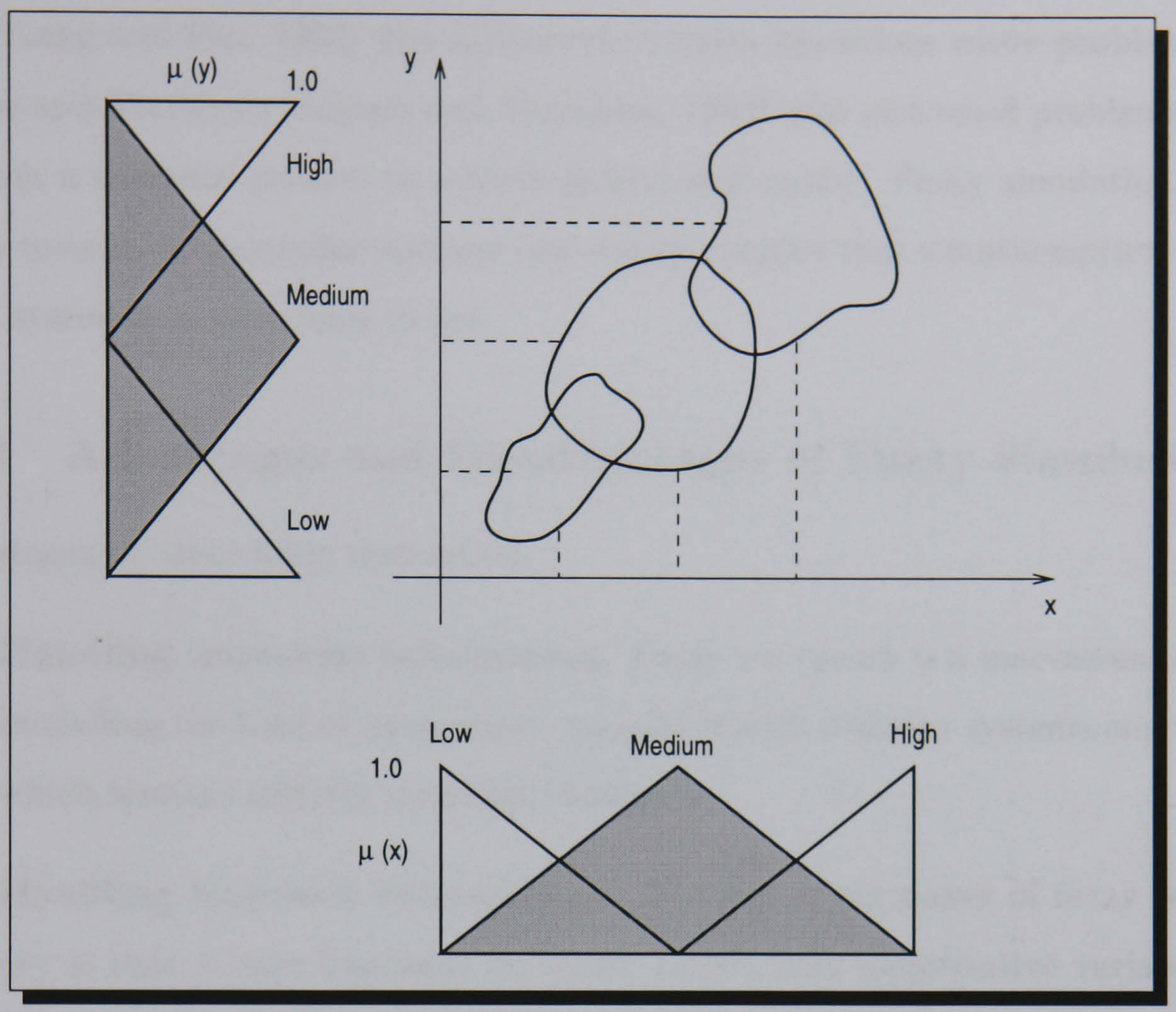


Figure 2.1: Fuzzy Associative Memories (FAMs)

The rule-based system represents a general nonlinear mapping from the input space of the fuzzy system to the output space of the fuzzy system. In this mapping, the patches of the input space are being applied to the patches in the output space. Each rule represents a fuzzy point of data that characterizes the nonlinear mapping from the input to the output.

There are many situations where only a complicated nonlinear process can be observed, whose functional relationship is not known, and whose behaviour is known only in the form of linguistic knowledge. The power of fuzzy nonlinear simulation is manifested in modelling nonlinear systems whose behaviour can be expressed in the form of input-output data-tuples, or in the form of linguistic rules of knowledge, and whose exact nonlinear specification is not known. Examples for fuzzy models that address such complex systems can be found in Huang and

Fan [Huang and Fan, 1993] who addressed complex hazardous waste problems and Sugeno and Yasukawa [Sugeno and Yasukawa, 1993] who addressed problems ranging from a chemical process to a stock price trend model. Fuzzy simulation is the ability to analyze dynamical systems that are so complex that a mathematical model is not available or very hard to get.

2.2.2 Advantages and Disadvantages of Fuzzy Simulation

Advantages of using fuzzy simulation:

- **Handling imprecise information.** Fuzzy set theory is a marvelous tool for modelling the kind of uncertainty associated with complex systems and issues, which humans address on a daily basis.

- **Handling linguistic information.** The underlying power of fuzzy set theory is that is uses linguistic variables, rather than quantitative variables, to represent imprecise concepts.

- **Extension principle of Zadeh [Zadeh, 1975a].** The extension principle provides a general method for extending crisp mathematical concepts to address fuzzy quantities, such as real algebra operations on fuzzy numbers.

- **Handling Complex Systems.** Fuzzy logic seems to be most successful in situations of handling very complex models where understanding is strictly limited or, in fact, quite judgemental.

- **Correct simulation results.** Given exact parameters it generates the same simulation results as classic mathematical computation.

Disadvantages of using fuzzy simulation:

- **Assignment problem of membership functions.** Membership functions essentially embody all fuzziness for a particular fuzzy set, its description

is the essence of a fuzzy property or operation. But the assignment process is the hard thing to do. There are methods, e.g. intuition, inference, rank ordering, neural networks [Takagi and Hayashi, 1991], generic algorithms [Karr and Gentry, 1993], etc., for overcoming the membership value assignment problem.

- **Loss of internal system structure.** Domain experts very often can model parts of the systems by basic functional constraints. These constraints represent parts of the system. The functional interplay of these parts builds the overall behaviour of the system. The knowledge about the internal structure of the system can not be modelled and is therefore lost.

- **No handling of incomplete information.** Information that is not available but needed for simulation, must be defined, usually by vaguely defined fuzzy sets.

- **Results need interpretation.** Like numerical simulation, fuzzy simulation produces detailed numerical output that describe the behaviour over time. To make sense of the output, a domain expert must interpret the output.

2.3 Historical Survey of Qualitative Fuzzy Simulation

2.3.1 Approximate Model-Based Reasoning

Naturally human thinking, reasoning, cognition, and perception is uncertain. According to Gupta, Knopf, and Nikiforuk [Gupta et al., 1988] uncertainty in cognitive information may arise due to the following three situations or combination of these:

Generality: This concept refers to a variety of possible situations of a phenomenon, that is, the defined universe is not just a point.

Ambiguity: This concept describes more than one distinguishable sub phenomena such that the membership will have several local maxima.

Vagueness: This concept reflects a set of phenomena with non-precise or non-crisp boundaries.

Central to all the reasoning styles in science of engineering is the creation of representations that capture the understanding that engineers or scientists have about physical phenomena. These representations comprise the physical knowledge for a domain. Physical knowledge is organized around models, i.e. structured descriptions of the phenomena of interest. A model of a system is structured into basic functional parts. The basic functional parts can be either a set of classical mathematical operations or propositions (see Chapter 7: "Imprecise Modelling").

In the traditional scientific approach a model consists of a finite set of formal relations. These mathematical relations among variables try to describe the real system. But for the design of analogue circuits it is, for example, difficult to find a small set of elementary laws to describe the interesting aspects of the observed phenomenon. Either the model might be too complex for its effective simulation or it is difficult to identify its parameters of a physical system not designed yet. In system design it is impossible to estimate with accuracy the parameter of a real system. The only available description, effectively usable in simulation may be an approximate model affected by uncertainty. The designer should be able to take into account the uncertainty and approximation involved in a complex, real system.

Approximate reasoning is defined by Zadeh in [Zadeh, 1975c] and [Zadeh, 1979] or by [Negoita, 1985] as reasoning with imprecise propositions. Approximate reasoning is analogous to predicate logic for reasoning with precise propositions, and hence is an extension of classical propositional calculus that deals with partial truths.

But the reasoning of this thesis is model-based. Neither qualitative reasoning as defined in Section 2.1 nor approximate reasoning is a suitable approach for the reasoning using imprecise models, parameters, and input data. Therefore approximate

model-based reasoning is defined as:

Definition 2.6 (*Approximate Model-Based Reasoning*): Reasoning is searching for models and predicting behaviours by qualitative fuzzy simulation.

The new reasoning definition has been introduced since there seemed to be none which described the reasoning adopted by this work. The reasoning is a fusion of the known *qualitative reasoning* and *approximate reasoning.*

2.3.2 Existing Approaches

Recently there have been attempts to overcome the disadvantages of qualitative and fuzzy simulation by combining both. Generally, in order to simulate mathematical models using the fuzzy set concept, three kinds of qualitative fuzzy simulation approaches have been reported:

1. **FUSIM (Fuzzy Simulator)** [Shen and Leitch, 1993] is based on Kuiper's QSIM [Kuipers, 1986]. This approach extends QSIM so that fuzzy qualitative values can be used. The qualitative values are mapped to fuzzy sets and the constraints and functional relations are described by fuzzy IF-THEN statements which are used to do fuzzy inference. All possible transitions from a qualitative state to its possible successors are generated. If there is more than one value for a variable the candidates are filtered to eliminate behaviours inconsistent with the relationships holding among variables. However, the set of next states may still contain a number of *spurious behaviours.*

2. **Fishwick's Methods** described in [Fishwick, 1991]

   (a) **Monte Carlo Method.** The Monte Carlo method is used with a fuzzy distribution instead of a probability distribution. In general, these methods are appropriate when enough samples can be obtained for an application so that uncertainty can be probabilistically specified as a density

distribution. When designing systems, it is impossible to gather statistical information because the system does not exist yet. Fishwick proposed to give fuzzy distributions as approximations of the unavailable probability distributions.

(b) **Correlated Uncertainty Method.** This method assumes that all uncertain sources are correlated. The simulation is numeric but results are still aggregated as fuzzy numbers at the end of n numeric simulations. Correlation is called in this work interaction of variables (see Chapter 5.5). Fuzzy numbers are represented as linear bounded L-R-Fuzzy Numbers (defined in 3.2). At each simulation step the previous simulation result is correlated to the next simulation result which can miss out some possible system behaviours.

(c) **Uncorrelated Uncertainty Method.** This method assumes that all the uncertainty sources are not correlated, i.e., that they are independent of one another. For this purpose, Fishwick uses fuzzy arithmetic in the numerical integration routine. Because of the divergent nature of the output at each step it makes this approach practically infeasible for most applications.

3. **QuaSi (Qualitative Simulation)** [Bonarini and Bontempi, 1994] Bonarini and Bontempi developed a simulator which accepts ODE-based and algebraic models which some or all of the parameters are approximately known. They proposed two different approaches: the non interacting approach and the interacting approach both discussed in detail in Chapter 6.5.2 and in Chapter 6.5.3 respectively.

All three approaches can simulate mathematical models although there is not enough information to simulate quantitatively.

2.4 Conclusion

Qualitative simulation is the attempt to handle incomplete and imprecise knowledge of physical systems. Soon there appeared approaches which overcame some of the drawbacks of the classical approaches of De Kleer and Brown [de Kleer and Brown, 1984], Forbus [Forbus, 1984], and Kuipers [Kuipers, 1984]. Most promising are approaches that add quantitative information because very often numerical information is available, too. Since quantitative information can also be expressed by fuzzy sets some approaches appeared which combined fuzzy logic and qualitative reasoning.

Both qualitative simulation and fuzzy simulation can handle imprecise information. Qualitative simulation is based on qualitative models represented by appropriate structure of a detailed model description and fuzzy simulation map input fuzzy sets to output fuzzy sets by IF-THEN rules.

Combining the qualitative and fuzzy approach is done in **FUSIM** [Shen and Leitch, 1993], **Fishwick's Methods** [Fishwick, 1991], and **QuaSi** [Bonarini and Bontempi, 1994]. Generally said they make it possible to simulate mathematical models although there is not enough information to simulate quantitatively.

Chapter 3

Representation of Imprecise Values

This chapter is more or less a review of value representation in fuzzy logic. Several well-known definitions are stated but necessary for chapters later on. All values, either qualitative, linguistic, fuzzy, or real, are represented by fuzzy sets which are defined by a characteristic function (membership function) over the crisp set of real numbers.

Qualitative reasoning makes use of a quantity space on which landmarks are defined. These landmarks are qualitative in nature similar to fuzzy values. The difference between them is that qualitative values are crisp, acting as symbols called semantics, while fuzzy values are not symbols but qualitative semantics represented by their membership functions. To unify the qualitative reasoning and fuzzy reasoning approach one of the two have to be mapped to the other approach. This thesis transfers the qualitative approach to the fuzzy approach by mapping the qualitative values to fuzzy sets.

3.1 Fuzzy Sets

In classical mathematical logic a real number x is a number of the real line with two possible truth values, true or false. Traditionally an element is or is not a member of a set.

Definition 3.7 (*Crisp Set*): Let X be a crisp (classical) set of objects, called the universe, whose generic elements are denoted x, $X = \{x\}$. A crisp set A over X is characterized by a membership function $\mu_{\tilde{A}}(x)$. This function classifies each element of X a membership value from the set $\{0, 1\}$, called "valuation set". 0 (1) is interpreted as false (true) respectively.

$\forall x \in X$:

$$\mu_A(x) = \begin{cases} 1, & if\ x \in A \\ 0, & if\ x \notin A \end{cases}$$

If the truth value of a number is true, it is certain that this number is correct, well-known or precise. If the truth value of a number is false, it is certain that exactly this number is not part of the used set.

Fuzzy logic takes into account that in everyday life there are things which are not known for certain if they are true or false. The truth value is an indicator for subjective information. This information is usually obtained from experience or from the opinion of the experts. Kaufmann and Gupta [Kaufmann and Gupta, 1991] defined this as a *level of presumption.*

If the value of a real number is exact but one wants to state that this value is not completely true, numerical truth values have to be used, which can be assigned with any value between false and true normalized to 0 and 1.

Definition 3.8 (*Level of Presumption*): The level of presumption is a measurement for the truth. How information is known for certain, if it is true or false.

Is the level of presumption 0 (1), information is known for certain that it is false (true) or not part of (part of) the fuzzy set.

Fuzzy set theory is a generalization of traditional (classical) set theory in the sense that the domain of the characteristic function is extended from the discrete set $0, 1$ to the closed real interval $[0, 1]$.

Definition 3.9 (*Fuzzy Set*): A fuzzy set $\tilde{A}$ of some universe X is represented by a generalized membership (characteristic) function from the universe X to the real unit interval $[0, 1]$, that is

$\mu_{\tilde{A}} : X \to [0, 1]$.

Kaufmann and Gupta [Kaufmann and Gupta, 1991] introduced the expression: *interval of confidence.*

Definition 3.10 (*Interval of Confidence*): The interval of confidence states the certainty of a number. It is certain that a number has a position between two other numbers.

In many cases an exact value is not possible to locate on the real line but it is possible to locate the value inside a closed interval of $\mathbb{R}$; that is, an interval of confidence of $\mathbb{R}$: $[r_1, r_2]$.

Compared to the level of presumption which is a subjective information the interval of confidence is objective information.

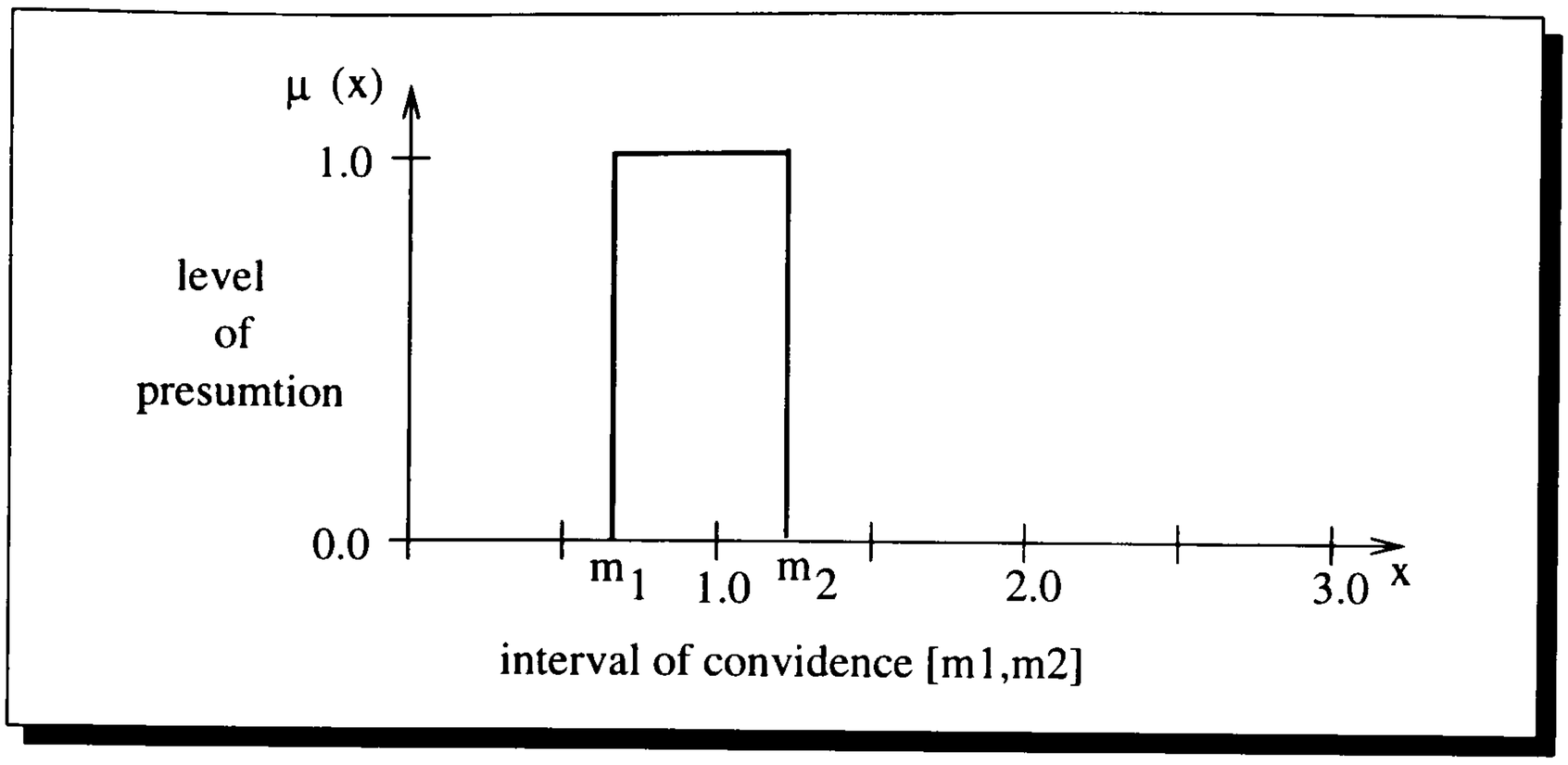


Figure 3.1: Crisp Real Interval in Fuzzy Logic

Fig.3.1 defines an interval of confidence in the fuzzy logic including the levels of presumption (truth value). Its membership function can be described by

$$\forall x, m_1, m_2 \in R:$$

$$\mu(x) = \begin{cases} 0, & if\ x < m_1 \\ 1, & if\ m_1 \leq x \leq m_2 \\ 0, & if\ m_2 < x \end{cases}$$

For numerical convenience the membership functions of the fuzzy sets are made discrete using α-cuts.

Definition 3.11 (α-*Cut*): A α-cut (α-level set, λ-cut) A_α of a fuzzy set $\tilde{A}$ and a real number $\alpha \in [0,1]$ is given by the following definition:

$$A_\alpha = \{x \in X | \mu_{\tilde{A}}(x) \geq \alpha\ with\ \alpha \in [0,1]\}$$

Fig. 3.2 shows an example of a fuzzy set $\tilde{A}$. Two α-cut set (intervals, crisp sets) $A_\alpha = [a3, a4]$ and $A_\beta = [a1, a2]$ are given at the two levels α and β.

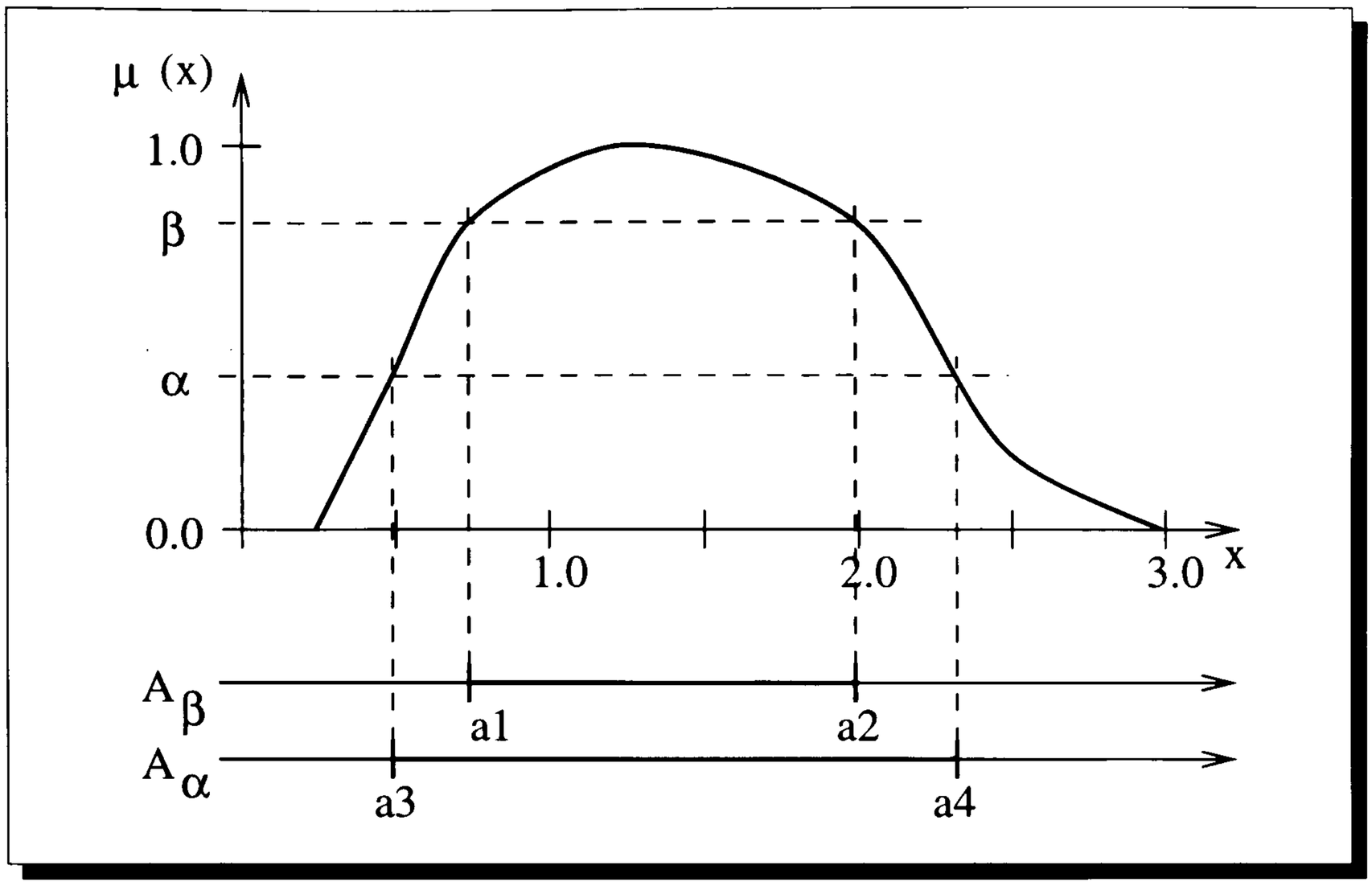


Figure 3.2: Two α-Cut Sets

3.2 Imprecise Numbers

The concept of a fuzzy number may be seen as one of the most fundamental concepts in fuzzy set theory. One of the most general definitions of a fuzzy quantity was made by Dubois and Prade in [Dubois and Prade, 1980]. Other definitions can be found by [Mizumoto and Tanaka, 1979] and [Kerre and van Schooten, 1988] in the book [Gupta and Yamakawa, 1988]. Generally fuzzy numbers can be defined as:

Definition 3.12 (*Fuzzy Number*): A fuzzy number is a convex normalized fuzzy set $\tilde{A}$ of the real line $\mathbb{R}$ such that

1. $\exists!\ x_0\ in\ \mathbb{R},\ \mu_{\tilde{A}}(x_0) = 1$ (x_0 is called the mean value of $\tilde{A}$);
2. $\mu_{\tilde{A}}$ is piecewise continuous.

Definition 3.13 (*Normal Fuzzy Set*): A normal fuzzy set $\tilde{A}$ is one whose membership function has at least one element x in the universe whose membership value is unity ($\mu_{\tilde{A}}(x) = 1$).

Definition 3.14 (*Convex Fuzzy Set*): A fuzzy set $\tilde{A}$ is said to be convex, if for all elements x, y, and z in a fuzzy set $\tilde{A}$, the relation $x < y < z$ implies that

$$\mu_{\tilde{A}}(y) \geq min[\mu_{\tilde{A}}(x), \mu_{\tilde{A}}(z)]$$

A common, more specific definition for fuzzy numbers is the following:

Definition 3.15 (*L-R Fuzzy Number*): A fuzzy set $\tilde{A}$ is called L-R fuzzy number if its membership function has the following form

$$\mu_{\tilde{A}}(x) = \begin{cases} L(\frac{m-x}{\alpha}), & if\ x \leq m,\ \alpha > 0 \\ R(\frac{x-m}{\beta}), & if\ x > m,\ \beta > 0 \end{cases}$$

with suitable characteristic functions $L(u)$ and $R(u)$ (e.g. $L(u) =:= R(u) = Max(0, 1 - u^2)$).

An important terminology is the support of a fuzzy set. The support of a fuzzy set is part of the input space for which its membership function is activated to a degree greater than zero.

Definition 3.16 (*Support*): The support of a fuzzy set $\tilde{A}$ is given by the following classical set S:

$$S(\tilde{A}) = \{x \in X | \mu_{\tilde{A}}(x) > 0\}.$$

Now compactness of a membership function can be discussed. Fig. 3.3 shows a non compact $(\tilde{A})$ and a compact $(\tilde{B})$ support fuzzy set.

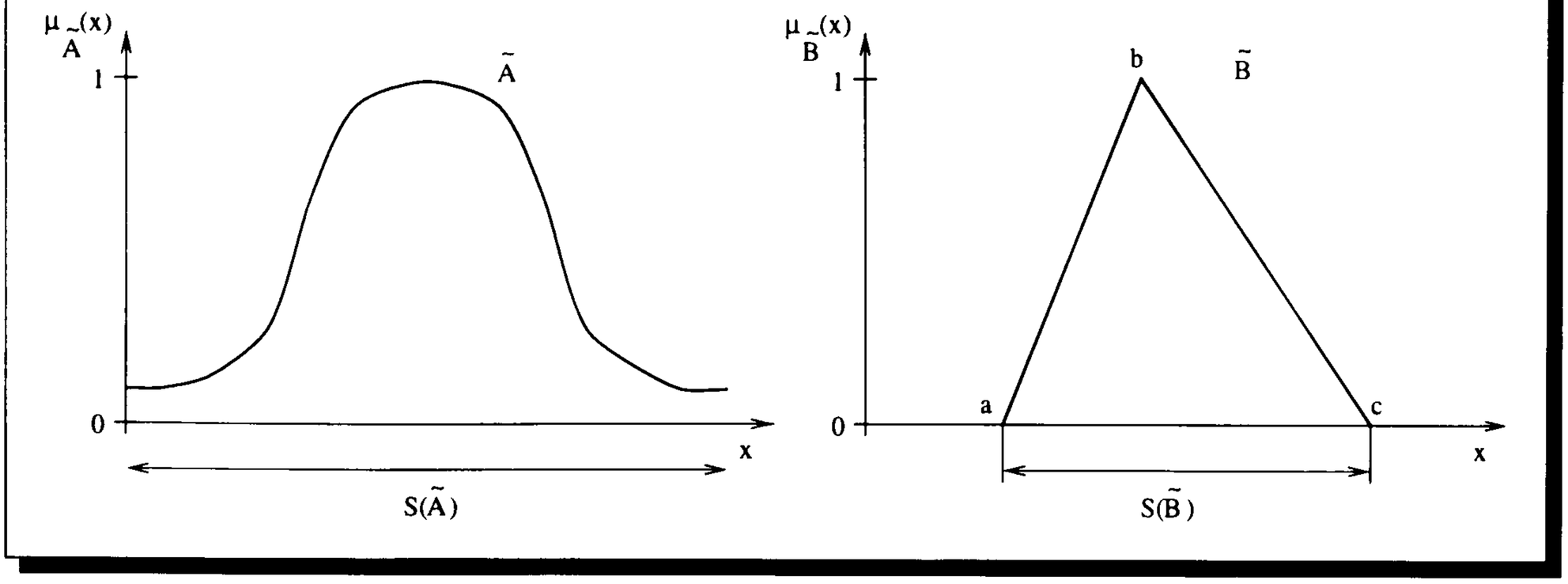


Figure 3.3: Non Compact Support (left fig.) and Compact Support (right fig.) Fuzzy Set

Compactness refers to the fact that the size of its support is strictly less than the size of the original universe of discourse. In a fuzzy rule-based system the fuzzy membership functions with a non-compact support are activated by each input thus the concept of local knowledge storage and retrieval may be lost. The compact support fuzzy set $\tilde{B}$ shown in Fig. 3.3 is defined by three values (a, b, c) and adequate for computational implementation. Therefore, a particular case of semi symmetric L-R fuzzy numbers is used, where the characteristic functions L and R are defined as follows:

$$L(u) = Max(0, 1-u) \text{ and } R(u) = Max(0, 1-u)$$

This specialization is known as *Triangular Fuzzy Numbers* (TFN).

Definition 3.17 (*Triangular Fuzzy Number (TFN)*): A fuzzy set $\tilde{A}$ is called triangular fuzzy number with one peak (or center) a, left width $\alpha > 0$ and right width $\beta > 0$ if its membership function has the following form

$$\mu_{\tilde{A}}(x) = \begin{cases} \frac{\alpha-(a-x)}{\alpha}, & if\ a-\alpha \leq x \leq a \\ \frac{\beta-(x-a)}{\beta}, & if\ a < x \leq b+\beta \\ 0, & otherwise \end{cases}$$

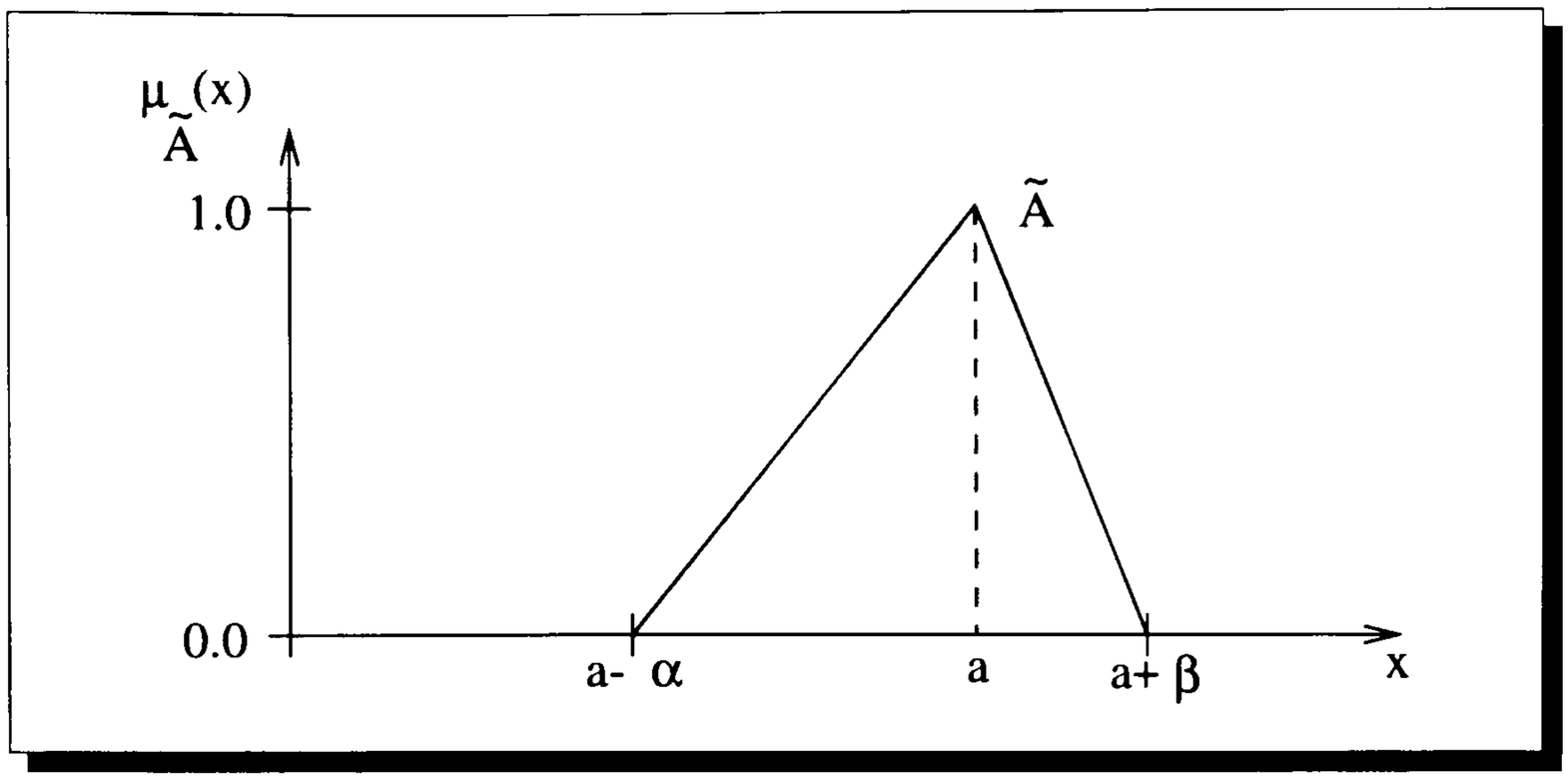


Figure 3.4: Triangular Fuzzy Number $\tilde{A}_{TFN} = (a, \alpha, \beta)$

and the notation $\tilde{A}_{triangular\ fuzzy\ number} =:= \tilde{A}_{TFN} = (a, \alpha, \beta)$ is used. If and only if α and β are *Zero* the value is an ordinary real number. Approximately 2 for x might be defined as: $\tilde{A} = (2, 1, 0.5)$ and is shown in Fig. 3.5

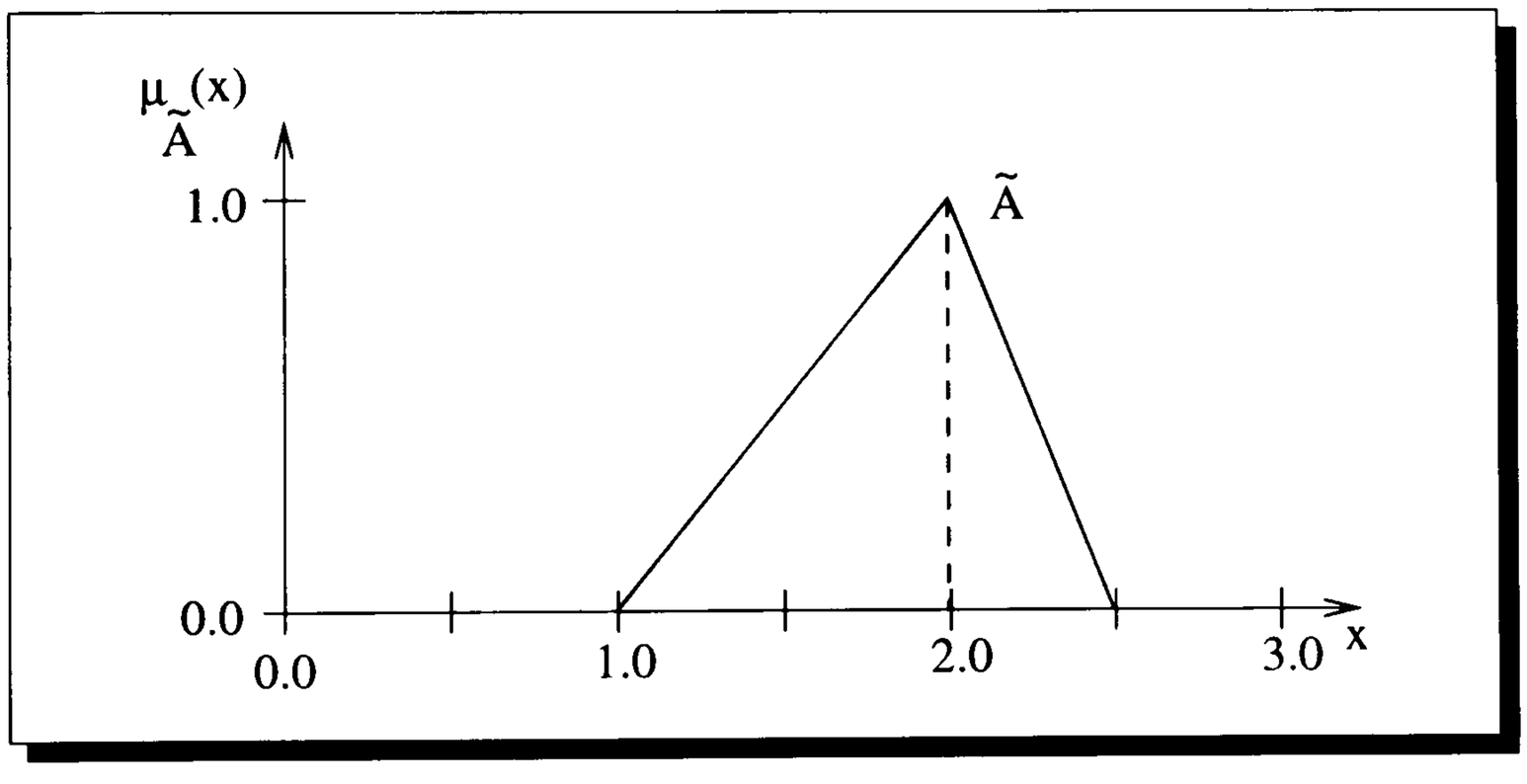


Figure 3.5: Approximate 2; Represented by $\tilde{A}_{TFN} = (2, 1, 0.5)$

3.3 Imprecise Intervals

A flat, triangular, or **TR**apezoidal, **F**uzzy **N**umber (TRFN), or **T**riangular **F**uzzy **I**nterval (TFI) will quite often arise from subjective expert opinion like "approxi-

mately in the interval".

Definition 3.18 (*Triangular Fuzzy Interval (TFI)*): A fuzzy set $\tilde{A}$ is called triangular fuzzy interval with the tolerance interval $[a, b]$, left width $\alpha > 0$ and right width $\beta > 0$ if its membership function has the following form

$$\mu_{\tilde{A}}(x) = \begin{cases} \frac{\alpha-(a-x)}{\alpha}, & if\ a-\alpha \leq x < a \\ 1, & if\ a \leq x \leq b \\ \frac{\beta-(x-b)}{\beta}, & if\ b < x \leq b+\beta \\ 0, & otherwise \end{cases}$$

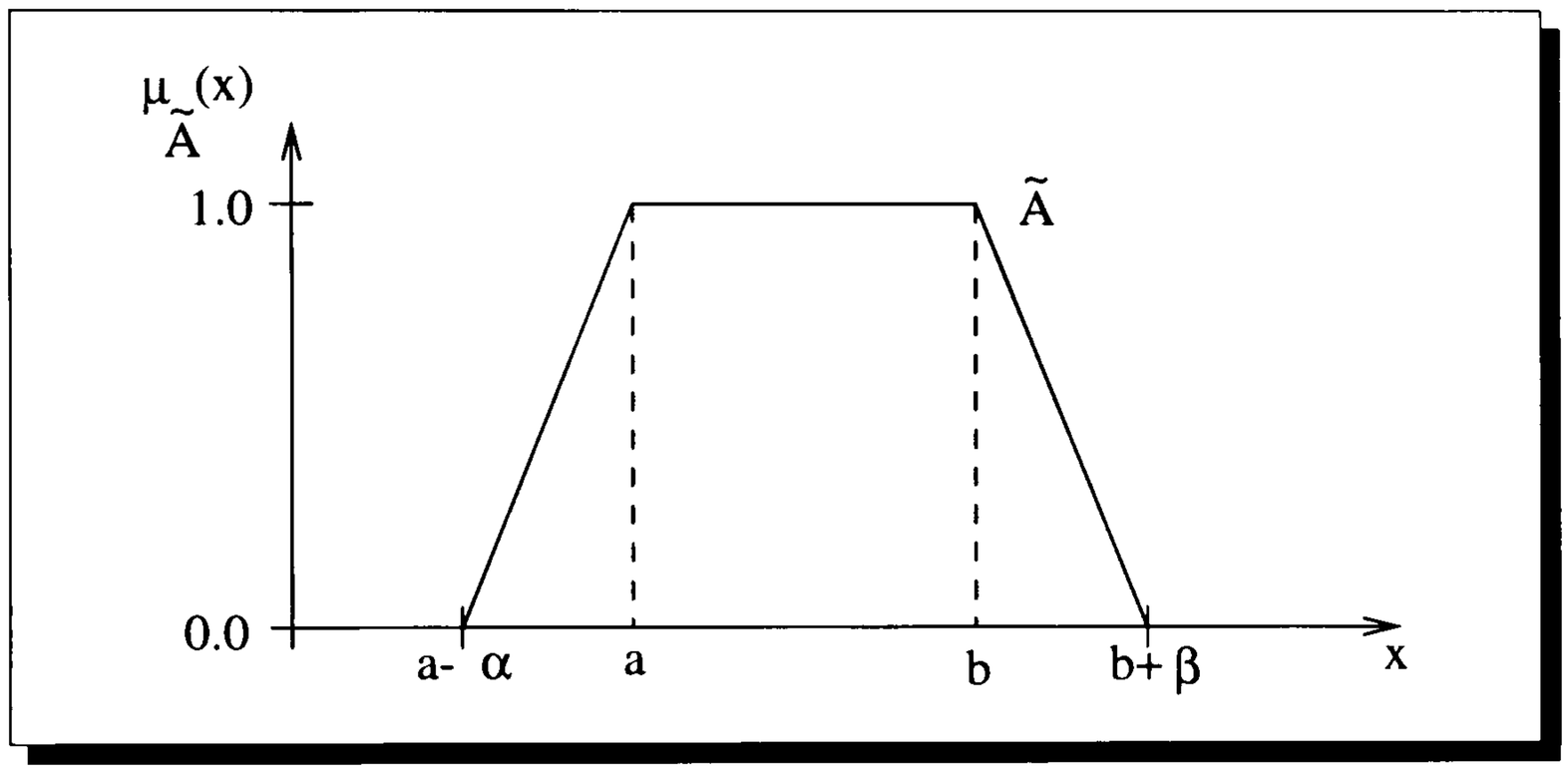


Figure 3.6: Triangular Fuzzy Interval $\tilde{A}_{TFI} = (a, b, \alpha, \beta)$

and the notation $\tilde{A}_{TFI} = (a, b, \alpha, \beta)$ is used.

Suppose a design engineer wants to specify that "x is approximately in the interval $[2, 3]$" (Fig. 3.7) using the notation above the triangular fuzzy interval $\tilde{A}_{TFI} = (2, 3, 1, 1.5)$ could be used.

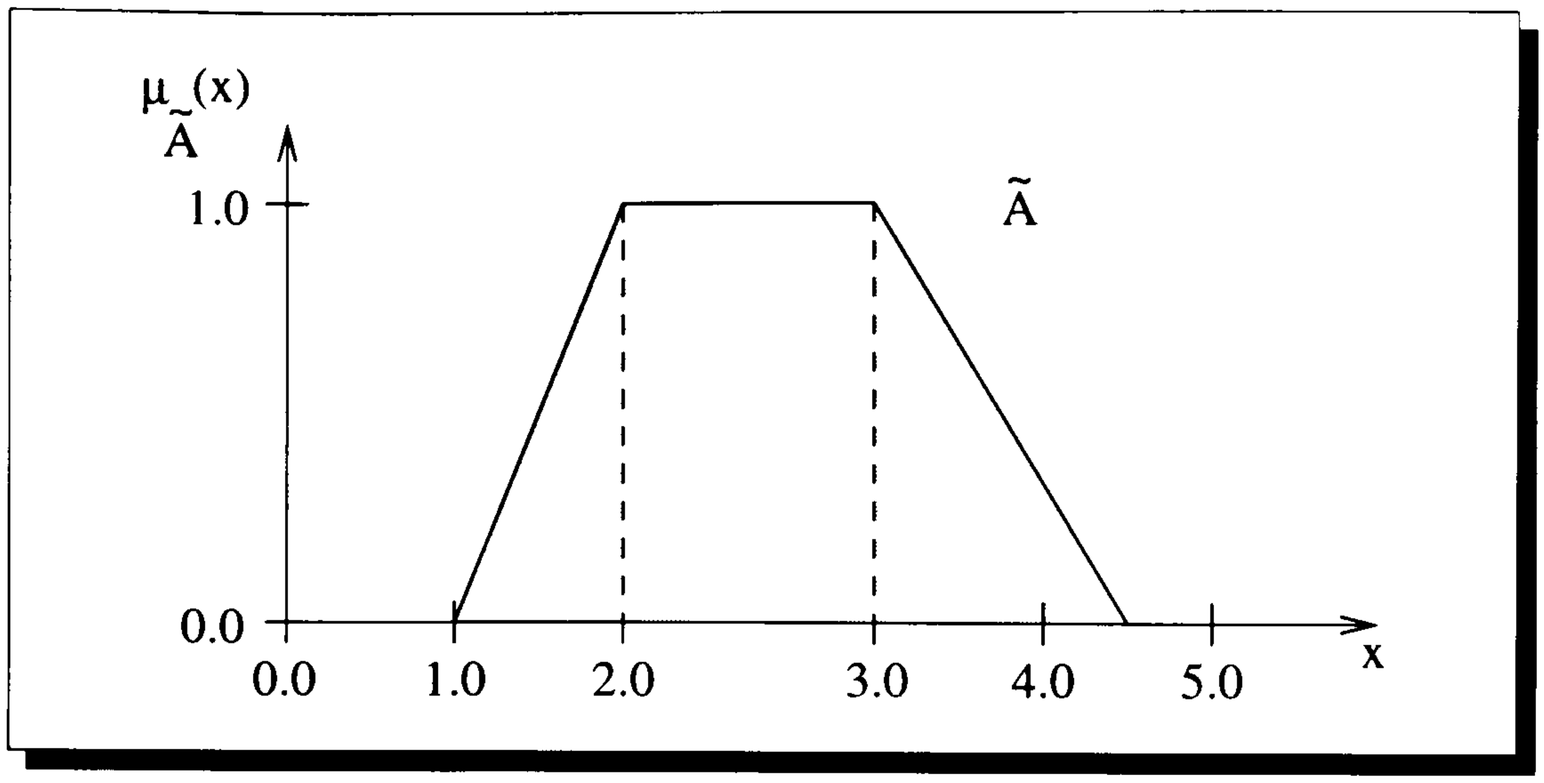


Figure 3.7: Approximate in $[2, 3]$; Represented by $\tilde{A}_{TFI} = (2, 3, 1, 1.5)$

3.4 Linguistic Variables, Linguistic Hedges

Natural language by itself is very often vague. Any attempt to rid our natural language of vagueness is unthinkable. Often domain knowledge engineers would like to describe some of the parameters especially when modelling rule-based of a nonlinear system using natural language. An essential characteristic of a vague predicate is that the boundaries of the domain of its applicability are not fixed, and, therefore, we do not know precisely where this domain ends and some other begins. The question of truth and falsity here is not only undecided but very often undecidable.

Of significance in practice are linguistic variables introduced by Zadeh [Zadeh, 1975a]. A linguistic variable differs from a numerical variable in that its values are not numbers but words or sentences in a natural or artificial language. Since words, in general, are less precise than numbers, the concept of a linguistic variable serves the purpose of providing a means of approximate characterization of phenomena which are too complex or too ill-defined to be amenable to description in conventional quantitative terms. The linguistic variable defined by Geyer-Schulz [Geyer-Schulz, 1995]:

Definition 3.19 (*Linguistic Variable*): A linguistic variable is a quintrupel $L = \{A, T(A), U, G, M\}$ with $A, T(A), U, G, M$ defined as follows:

- A is the name of the linguistic variable.
- $T(A)$ is the domain of verbal values of A.
- Every single value of A named Y_i represents a fuzzy set over the base set U.
- G is the definition of the grammar that produces the names of the values of A.
- U is a semantic rule, that maps every verbal value of A to a meaning $M(Y_i)$, i.e. to a fuzzy subset of the base set.

Suppose we want to specify a linguistic variable for the room temperature $A =$ *room temperature*, then $T(A) = \{low,\ normal,\ high\}$. The base set U is defined as $U = [0, 60]^{\circ}C$. The membership functions for *low*, *normal*, *high* are defined by Fig. 3.8

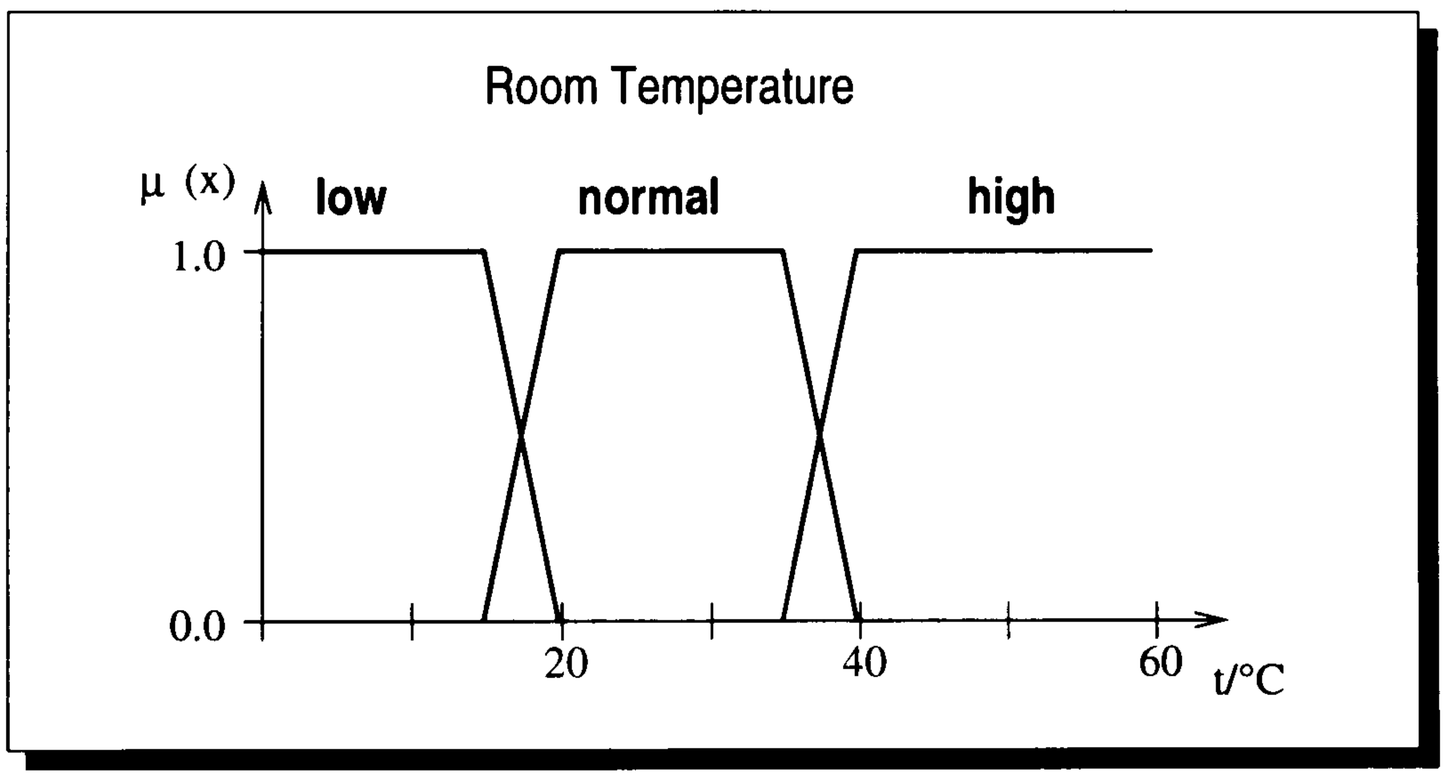


Figure 3.8: Membership Functions for the Linguistic Variable: *room temperature*

Lotfi A. Zadeh [Zadeh, 1975a] introduced linguistic truth values, e.g. "very true", "true", "fairly true", etc. Such truth values may be regarded as fuzzy sets on the unit interval that is characterized by its own membership function.

In linguistics, fundamental atomic terms are often modified with adjectives or adverbs like very, low, slightly, fairly, etc. These are called modifiers (linguistic hedges), that is, the singular meaning of an atomic term is modified, or hedged, from its original interpretation. Linguistic hedges have the effect of modifying the membership function for a basic atomic term, e.g. Likelihood, such as "likely", "very likely", "highly likely", "unlikely".

Neither linguistic truth values nor linguistic hedges have been further investigated in this thesis. For the theory of approximate model-based reasoning approach it is more important that they can be handled as ordinary fuzzy sets and modifiers of fuzzy sets.

3.5 Representation of Qualitative Values

The starting point for qualitative reasoning is the definition of the qualitative domain Q ($\mathbb{Q}$-domain) also called quantity space.

Definition 3.20 (*Qualitative Domain*): A qualitative domain is a finite, totally ordered set of symbols, the landmark values.

Landmarks are essential in qualitative simulation and they are part of the representation of qualitative values.

The most general, historically most widely used $\mathbb{Q}$-domain is $Q_R = \{\boxed{-}, \boxed{O}, \boxed{+}\}$. This $\mathbb{Q}$-domain consists of one landmark $\boxed{O}$. $\boxed{O}$ divides the positive numbers $\boxed{+}$ from the negative numbers $\boxed{-}$ on the real number. Real numbers must be mapped to the limited set Q of qualitative values, since information in the form of real numbers is not useful or necessary. The quantities of the real numbers are mapped to the quantities of the qualitative numbers $R \rightarrow Q_R$.

$$[\,] : R \rightarrow Q_R, [x] := \begin{cases} \boxed{+} & if\ x > \epsilon \\ \boxed{O} & if\ |x| \leq \epsilon; \epsilon \in R \\ \boxed{-} & if\ x < -\epsilon \end{cases}$$

The real number ϵ allows the design engineer to define at which point in the analogue circuit design domain something is zero.

Definition 3.21 (*Qualitative Landmark*): A qualitative landmark value is a symbolic name for a particular real number, whose numerical value is known or not. Landmarks break a continuous set of values into qualitatively distinct regions.

Definition 3.22 (*Qualitative Interval*): A qualitative interval is defined as the region between two landmarks.

As stated in the introduction of this chapter it is necessary to map qualitative values to the fuzzy approach by representing both, *qualitative landmarks* and *qualitative intervals* by fuzzy sets.

Landmark values whose numerical values are not known are very vague representation of values. An engineer should be able to put these landmark values on a firmer ground and represent them as fuzzy numbers. To do qualitative fuzzy simulation the qualitative values have to be mapped to fuzzy values. Therefore landmarks are represented by fuzzy numbers and called *fuzzy landmarks.*

Definition 3.23 (*Fuzzy Landmark*): A fuzzy landmark value is a symbolic name for a particular fuzzy number.

When landmarks of the qualitative interval are no particular point they are represented by *fuzzy qualitative intervals.*

Definition 3.24 (*Fuzzy Qualitative Interval*): A fuzzy qualitative interval value is a symbolic name for a particular fuzzy interval. It consists of a minimum and maximum value, with their degree of imprecision.

The qualitative domain defined by the fuzzy landmarks can be seen as a linguistic variable whose landmarks are the verbal values of the linguistic variable. Suppose the qualitative domain for the *water temperature* is defined as following:

$$Q = \{cold, warm, hot\}$$

A possible mapping of the qualitative domain to a qualitative fuzzy quantity space can be seen in Fig. 3.9.

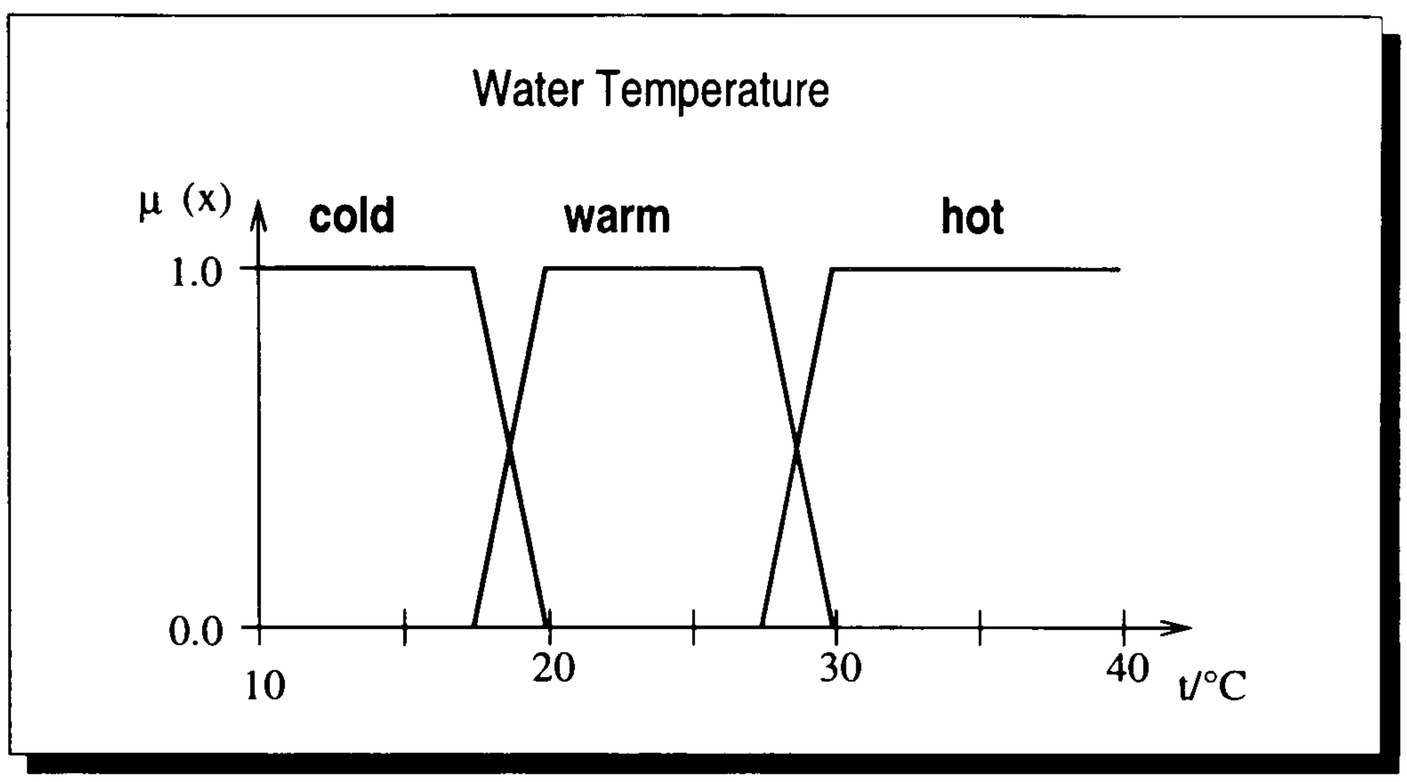


Figure 3.9: Qualitative Fuzzy Quantity Space for the Qualitative Domain: *water temperature*

Each single fuzzy landmark is defined by a membership function, as following:

$$\mu_{cold}(temperature) = \begin{cases} 1, & if\ temperature \leq 17.5^{\circ}C \\ \frac{20-temperature}{2.5}, & if\ 17.5^{\circ}C < temperature \leq 20^{\circ}C \\ 0, & if\ temperature > 20^{\circ}C \end{cases}$$

$$\mu_{warm}(temperature) = \begin{cases} 0, & if\ temperature \leq 17.5^{\circ}C \\ \frac{temperature-17.5}{2.5}, & if\ 17.5^{\circ}C < temperature < 20^{\circ}C \\ 1, & if\ 20^{\circ}C \leq temperature \leq 27.5^{\circ}C \\ \frac{30-temperature}{2.5}, & if\ 27.5^{\circ}C < temperature \leq 30^{\circ}C \\ 0, & if\ temperature > 30^{\circ}C \end{cases}$$

$$\mu_{hot}(temperature) = \begin{cases} 0, & if\ temperature \leq 27.5^{\circ}C \\ \frac{temperature-27.5}{2.5}, & if\ 27.5^{\circ}C < temperature < 30^{\circ}C \\ 1, & if\ 30^{\circ}C \leq temperature \end{cases}$$

3.6 Conclusion

Part of the knowledge and input data of a nonlinear system is static during the simulation process — they do not change their value. This particular kind of knowledge, often imprecise, can be expressed by:

- fuzzy numbers, e.g. temperature is approximately 30°C,
- fuzzy intervals, e.g. temperature is approximately between 20°C and 40°C,
- linguistic values, e.g. temperature is warm,
- linguistic hedges, e.g. temperature is very warm,
- fuzzy landmarks, e.g. temperature is room temperature,
- fuzzy qualitative intervals, e.g. temperature is between freeze and boil temperature.

Usually a fuzzy number, or fuzzy intervals do not need to have a linguistic interpretation. Linguistic terms and linguistic hedges are used to ease the definition and interpretation of the membership functions and the qualitative domain. The qualitative domain can be interpreted as a linguistic variable whereas A is the qualitative domain, and the landmarks of that qualitative domain are the $T(A)$, the set names of the qualitative domain.

Chapter 4

Representation of Imprecise Analogue Signals

Continuous signals are used to represent dynamic information by physical quantities. These quantities are typically, for example analogue signals [1], voltage or current data of analogue circuits which change their value during simulation. Of special interest are analogue signals varying in time. These signals, used as model parameters, are not known exactly during the design process. A design engineer specifies such imprecise signals most easily by a drawing tool described in Chapter 10. Internally imprecise analogue signals are represented by the new fuzzy type **Fuzzy Relation Memories** — Fuzzy Curves introduced by Reich ([Reich, 1997b] and [Reich, 1998]). This chapter describes the development of fuzzy curves from fuzzy relations through fuzzifying functions to Fuzzy Relation Memories. This chapter discusses the use of FRMs and shows how the ordering problem (defined in Section 2.1.2) of qualitative reasoning is solved by using FRMs.

[1] It is called an analogue signal if the quantity can reach any value in between limits.

4.1 Fuzzy Functions — Fuzzy Relations (FR)

An imprecise linear function can be represented by a fuzzy relation.

Definition 4.25 (*Fuzzy Relation*): A fuzzy relation is defined as:

$$\tilde{y} = (\tilde{a} \odot x) \oplus \tilde{b} \text{ with } x \in X \tag{4.1}$$

where $\tilde{a}$ and $\tilde{b}$ are fuzzy intervals and $\odot$ and $\oplus$ are the extended operations multiplication and addition (see Appendix A: *Fuzzy Reasoning Basics*) respectively on fuzzy sets. The result $\tilde{y}$ of this fuzzy relation is again a fuzzy interval.

In the following, only fuzzy intervals with trapezoidal membership functions will be used. The membership functions are restricted to trapezoidal membership functions, because:

- it is computationally less expensive then nonlinear boundaries. Just two points of a linear boundary is enough to define the function.
- the accuracy representing the fuzzy sets with linear boundaries is adequate in the domain of design. During the design process the membership functions are defined by intuition (see 10.3.1). This subjective definition of the design parameters does not need higher accuracy.

Suppose a fuzzy set is represented by a set of α-level sets $A_\alpha = \{x \in X | \mu_{\tilde{A}}(x) \geq \alpha\}$. Then the fuzzy relation above can be seen as a set of linear functions associated to a particular α-level for

$$\begin{aligned} \text{the upper bounds: } & f_\alpha^+(x) = y_\alpha = a_\alpha^+ * x + b_\alpha^+ \; with \; \alpha \in [0,1] \\ \text{the lower bounds: } & f_\alpha^-(x) = y_\alpha = a_\alpha^- * x + b_\alpha^- \; with \; \alpha \in [0,1] \end{aligned} \tag{4.2}$$

If two fuzzy numbers are multiplied/divided the left and right boundary of the multiplication/division are of quadratic nature. For numerical convenience the boundary of the evaluated fuzzy values are linearized discussed in Appendix A: *Fuzzy Reasoning Basics*.

Definition 4.26 (*Membership Function of Fuzzy Relation*): The membership function for the fuzzy relation 4.1 for fuzzy interval $\tilde{y}$ is defined as:

$\forall x, x \geq 0$:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq (a^-_{>0.0} * x + b^-_{>0.0}) \\ \frac{y-(a^-_{>0.0}*x+b^-_{>0.0})}{(a^-_{1.0}-a^-_{>0.0})*x+b^-_{1.0}-b^-_{>0.0}} & if \quad (a^-_{>0.0} * x + b^-_{>0.0}) < y \leq (a^-_{1.0} * x + b^-_{1.0}) \\ 1 & if \quad (a^-_{1.0} * x + b^-_{1.0}) < y < (a^+_{1.0} * x + b^+_{1.0}) \\ \frac{(a^+_{>0.0}*x+b^+_{>0.0})-y}{(a^+_{>0.0}-a^+_{1.0})*x+b^+_{>0.0}-b^+_{1.0}} & if \quad (a^+_{1.0} * x + b^+_{1.0}) \leq y < (a^+_{>0.0} * x + b^+_{>0.0}) \\ 0 & if \quad (a^+_{>0.0} * x + b^+_{>0.0}) \leq y \end{cases} \tag{4.3}$$

with:

$$\begin{aligned} f^-_{>0.0}(x) &= a^-_{>0.0} * x + b^-_{>0.0} \\ f^-_{1.0}(x) &= a^-_{1.0} * x + b^-_{1.0} \\ f^+_{1.0}(x) &= a^+_{1.0} * x + b^+_{1.0} \\ f^+_{>0.0}(x) &= a^+_{>0.0} * x + b^+_{>0.0} \end{aligned} \tag{4.4}$$

$\forall x, x < 0$:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq (a^+_{>0.0} * x + b^-_{>0.0}) \\ \frac{y-(a^+_{>0.0}*x+b^-_{>0.0})}{(a^+_{1.0}-a^+_{>0.0})*x+b^-_{1.0}-b^-_{>0.0}} & if \quad (a^+_{>0.0} * x + b^-_{>0.0}) < y \leq (a^+_{1.0} * x + b^-_{1.0}) \\ 1 & if \quad (a^+_{1.0} * x + b^-_{1.0}) < y < (a^-_{1.0} * x + b^+_{1.0}) \\ \frac{(a^-_{>0.0}*x+b^+_{>0.0})-y}{(a^-_{>0.0}-a^-_{1.0})*x+b^+_{>0.0}-b^+_{1.0}} & if \quad (a^-_{1.0} * x + b^+_{1.0}) \leq y < (a^-_{>0.0} * x + b^+_{>0.0}) \\ 0 & if \quad (a^-_{>0.0} * x + b^+_{>0.0}) \leq y \end{cases} \tag{4.5}$$

with:

$$\begin{aligned} f^-_{>0.0}(x) &= a^+_{>0.0} * x + b^-_{>0.0} \\ f^-_{1.0}(x) &= a^+_{1.0} * x + b^-_{1.0} \\ f^+_{1.0}(x) &= a^-_{1.0} * x + b^+_{1.0} \\ f^+_{>0.0}(x) &= a^-_{>0.0} * x + b^+_{>0.0} \end{aligned} \tag{4.6}$$

4.1.1 Example: Fuzzy Relation

Fig. 4.1 shows a fuzzy relation $\tilde{y} = (\tilde{a} \odot x) \oplus \tilde{b}$ with $\tilde{a} = (0.8, 1.2, 0.2, 0.2)$ and $\tilde{b} = (4, 5, 1, 1)$.

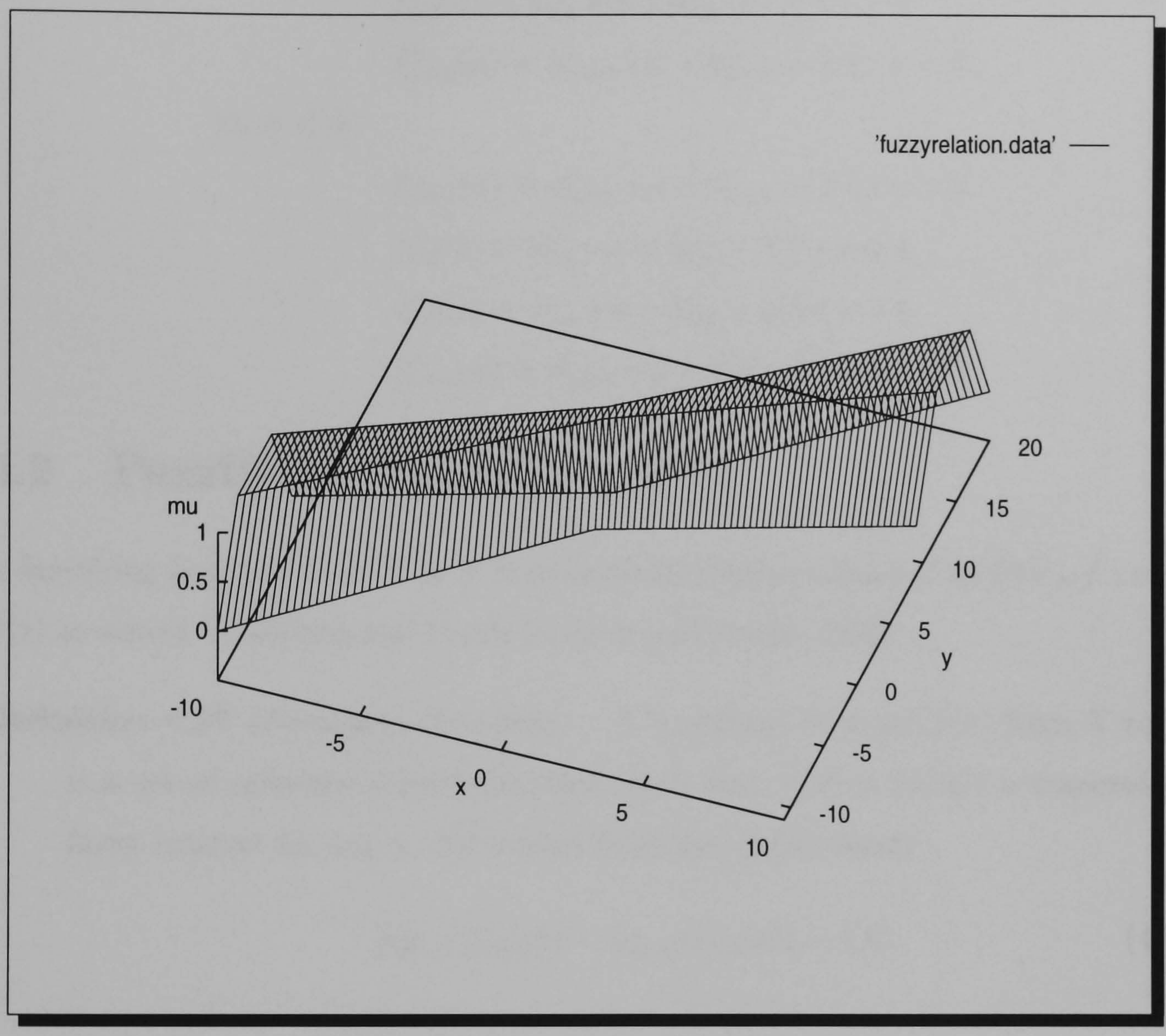


Figure 4.1: Fuzzy Relation $\tilde{y} = (\tilde{a} \odot x) \oplus \tilde{b}$ with $\tilde{a} = (0.8, 1.2, 0.2, 0.2)$ and $\tilde{b} = (4, 5, 1, 1)$

In the example Fig. 4.1 it can be seen that 8 functions at the α-cut level $= \{> 0.0, 1.0\}$ could completely describe the fuzzy relation.

$\forall x, x \geq 0$:

$$f^{-}_{>0.0}(x) = a^{-}_{>0.0} * x + b^{-}_{>0.0} = 0.6 * x + 3$$
$$f^{-}_{1.0}(x) = a^{-}_{1.0} * x + b^{-}_{1.0} = 0.8 * x + 4$$
$$f^{+}_{1.0}(x) = a^{+}_{1.0} * x + b^{+}_{1.0} = 1.2 * x + 5$$
$$f^{+}_{>0.0}(x) = a^{+}_{>0.0} * x + b^{+}_{>0.0} = 1.4 * x + 6$$

$\forall x, x < 0$:

$$f^{-}_{>0.0}(x) = a^{+}_{>0.0} * x + b^{-}_{>0.0} = 1.4 * x + 3$$
$$f^{-}_{1.0}(x) = a^{+}_{1.0} * x + b^{-}_{1.0} = 1.2 * x + 4$$
$$f^{+}_{1.0}(x) = a^{-}_{1.0} * x + b^{+}_{1.0} = 0.8 * x + 5$$
$$f^{+}_{>0.0}(x) = a^{-}_{>0.0} * x + b^{+}_{>0.0} = 0.6 * x + 6$$

4.2 Fuzzifying Functions (FF)

A fuzzifying function from X to Y is an ordinary function from X to $\tilde{P}(Y)$, $\tilde{f} : x \mapsto \tilde{f}(x)$ as stated by Dubois and Prade [Dubois and Prade, 1980].

Definition 4.27 (*Fuzzifying Function*): A fuzzifying function $\tilde{f}(x)$ from X to Y is a set of ordinary α-level functions such that $\tilde{f}(x)$ is always a trapezoidal fuzzy interval for any x. All α-level functions $f_\alpha(x)$ satisfy

$$\mu_{\tilde{f}(x)}(f^{-}_{1.0}(x)) = \mu_{\tilde{f}(x)}(f^{+}_{1.0}(x)) = 1.0 \tag{4.7}$$

$$\mu_{\tilde{f}(x)}(f^{-}_{\alpha}(x)) = \mu_{\tilde{f}(x)}(f^{+}_{\alpha}(x)) = \alpha \; with \; \alpha \in [0, 1] \tag{4.8}$$

The membership function for $\tilde{y} = \tilde{f}(x)$ with
the α-cut-level-functions= $f_{\alpha_1}, f_{\alpha_2}, f_{\alpha_3}, ..., f_{\alpha_n}$, ($\alpha_1$ => 0.0 and $\alpha_n = 1.0$) is

defined as:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} \alpha_1 & if \quad y \le f^-_{\alpha_1}(x) \\ \frac{y*(\alpha_1-\alpha_2)+\alpha_2*f^-_{\alpha_1}(x)-\alpha_1*f^-_{\alpha_2}(x)}{f^-_{\alpha_1}(x)-f^-_{\alpha_2}(x)} & if \quad f^-_{\alpha_1}(x) < y \le f^-_{\alpha_2}(x) \\ \frac{y*(\alpha_2-\alpha_3)+\alpha_3*f^-_{\alpha_2}(x)-\alpha_2*f^-_{\alpha_3}(x)}{f^-_{\alpha_2}(x)-f^-_{\alpha_3}(x)} & if \quad f^-_{\alpha_2}(x) < y \le f^-_{\alpha_3}(x) \\ . & \\ . & \\ . & \\ \alpha_n & if \quad f^-_{\alpha_n}(x) < y < f^+_{\alpha_n}(x) \\ . & \\ . & \\ . & \\ \frac{y*(\alpha_2-\alpha_3)+\alpha_3*f^+_{\alpha_2}(x)-\alpha_2*f^+_{\alpha_3}(x)}{f^+_{\alpha_2}(x)-f^+_{\alpha_3}(x)} & if \quad f^+_{\alpha_3}(x) \le y < f^+_{\alpha_2}(x) \\ \frac{y*(\alpha_1-\alpha_2)+\alpha_2*f^+_{\alpha_1}(x)-\alpha_1*f^+_{\alpha_2}(x)}{f^+_{\alpha_1}(x)-f^+_{\alpha_2}(x)} & if \quad f^+_{\alpha_2}(x) \le y < f^+_{\alpha_1}(x) \\ \alpha_1 & if \quad f^+_{\alpha_1}(x) \le y \end{cases} \tag{4.9}$$

Hence it follows that the functions $f^-_\alpha(x)$, $f^-_{1.0}(x)$, $f^+_{1.0}(x)$, and $f^+_\alpha(x)$ completely define a fuzzifying functions (Fig.4.2).

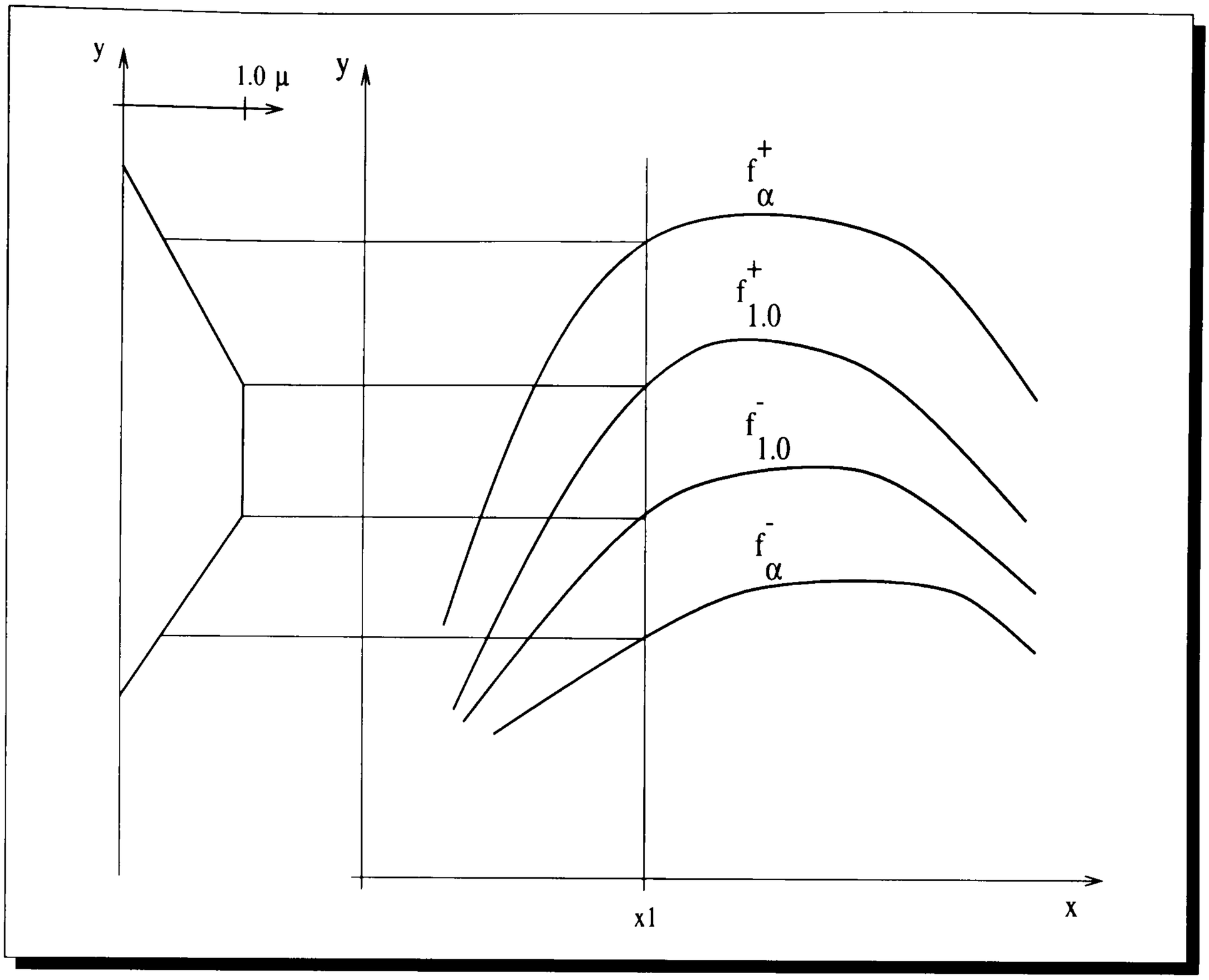


Figure 4.2: Fuzzifying Function

Suppose only the two α-levels, $\alpha > 0$ and $\alpha = 1.0$, are defined. This restricts the fuzzifying functions to fuzzy intervals and linear boundaries following the membership function for $\tilde{y} = \tilde{f}(x)$ as:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq f^{-}_{>0.0}(x) \\ \frac{y - f^{-}_{>0.0}(x)}{f^{-}_{1.0}(x) - f^{-}_{>0.0}(x)} & if \quad f^{-}_{>0.0}(x) < y \leq f^{+}_{1.0}(x) \\ 1 & if \quad f^{-}_{1.0}(x) < y < f^{+}_{1.0}(x) \\ \frac{f^{+}_{>0.0}(x) - y}{f^{+}_{>0.0}(x) - f^{+}_{1.0}(x)} & if \quad f^{+}_{1.0}(x) \leq y < f^{+}_{>0.0}(x) \\ 0 & if \quad f^{+}_{>0.0}(x) \leq y \end{cases} \tag{4.10}$$

4.2.1 Example: Fuzzifying Function

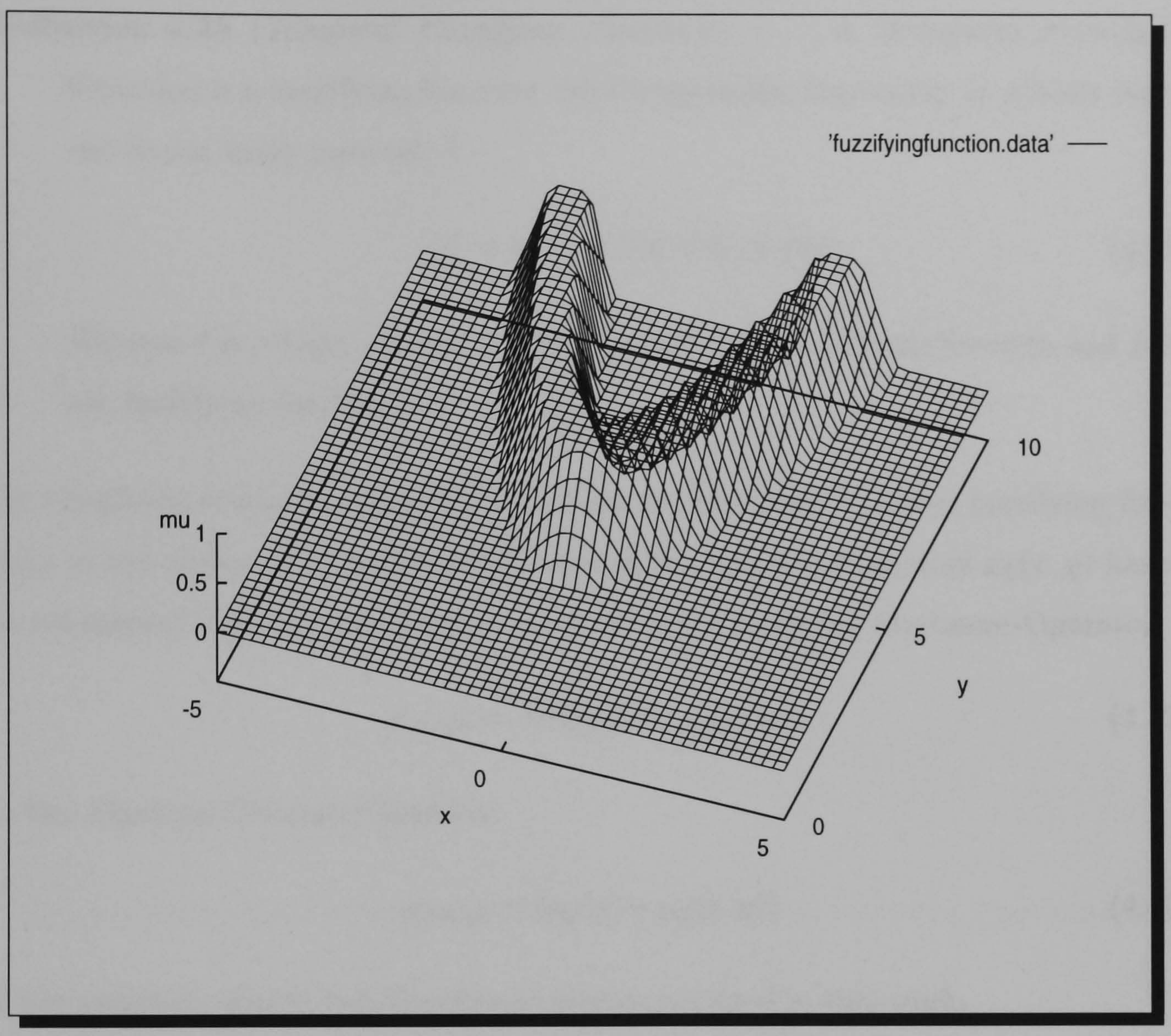


Figure 4.3: Example: Fuzzifying Function

Fig. 4.3 shows a fuzzifying function with the following α-level functions:

$$f^{-}_{>0.0}(x) = 0.6 * x^2 + 3 \qquad f^{-}_{1.0}(x) = 0.8 * x^2 + 4$$
$$f^{+}_{1.0}(x) = 1.2 * x^2 + 5 \qquad f^{+}_{>0.0}(x) = 1.4 * x^2 + 6$$

4.3 Temporal Fuzzifying Functions (TFF)

Definition 4.28 (*Temporal Fuzzifying Function*): A Temporal Fuzzifying Function is a fuzzifying function valid temporally, depending on a fuzzy interval (input fuzzy interval) $\tilde{I}$.

$$IF\ x\ is\ I,\ THEN\ \tilde{y}\ is\ f(x) \tag{4.11}$$

whereas $\tilde{I}$ is a fuzzy interval with a trapezoidal membership function and $\tilde{f}(x)$ are fuzzifying functions.

For simplicity temporal fuzzifying functions are restricted to linear fuzzifying functions in this thesis. The membership value of the fuzzifying function $\mu_{\tilde{y}}(x,y)$ has to be considered with the membership value of $\mu_{\tilde{I}}(x)$ using the Minimum-Operator

$$\mu_{result} = min[\mu_{\tilde{I}}(x), \mu_{\tilde{y}}(x,y)] \tag{4.12}$$

or the Algebraic-Product-Operator

$$\mu_{result} = [\mu_{\tilde{I}}(x) * \mu_{\tilde{y}}(x,y)] \tag{4.13}$$

If not explicitly stated the Minimum-Operator is used in this work.

4.3.1 Example: Temporal Fuzzifying Function

Fig. 4.4 shows an example with $\tilde{I} = (2,4,1,1)$ and linearly defined fuzzifying function with the following α-level functions:

$$f^{-}_{>0.0}(x) = 0.6 * x + 3 \quad f^{-}_{1.0}(x) = 0.8 * x + 4$$
$$f^{+}_{1.0}(x) = 1.2 * x + 5 \quad f^{+}_{>0.0}(x) = 1.4 * x + 6$$

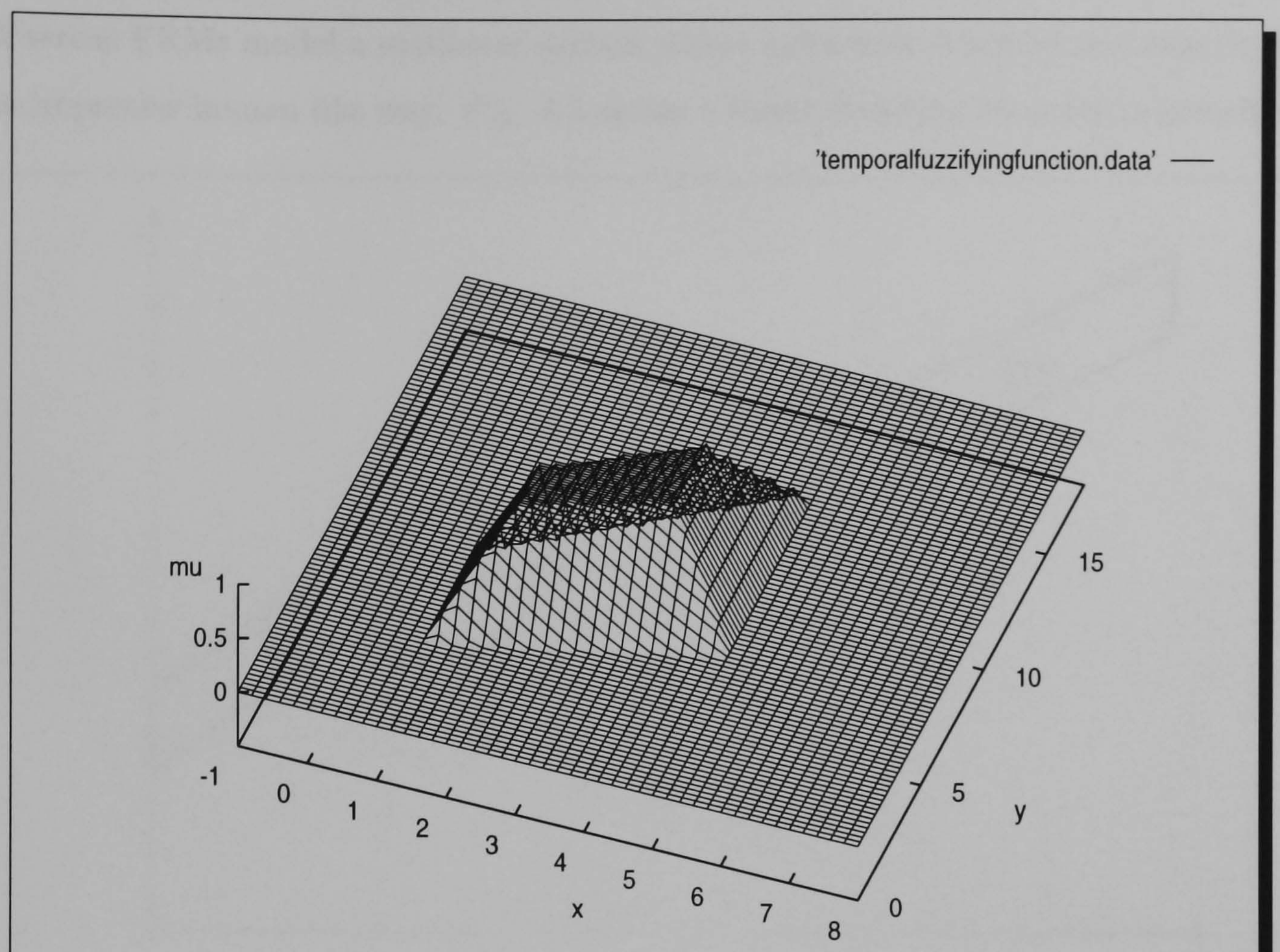


Figure 4.4: Temporal Fuzzifying Function

4.4 Fuzzy Relation Memories (FRMs) — Fuzzy Curves

Fuzzy Relation Memories are based on Fuzzy Association Memories (FAMs) as stated in B. Kosko [Kosko, 1994]. An imprecise function is approximated by covering its graph with temporal fuzzifying functions. The FAMs describe their input and output restrictions by fuzzy sets; FRMs describe their input restrictions by fuzzy sets and their output restrictions by fuzzifying functions. FAMs model a nonlinear system whose behaviour is known exactly by an imprecise human like way.

Whereas FRMs model a nonlinear system whose behaviour is not known exactly by an imprecise human like way. Fig. 4.5 shows a Fuzzy Relation Memory in principle.

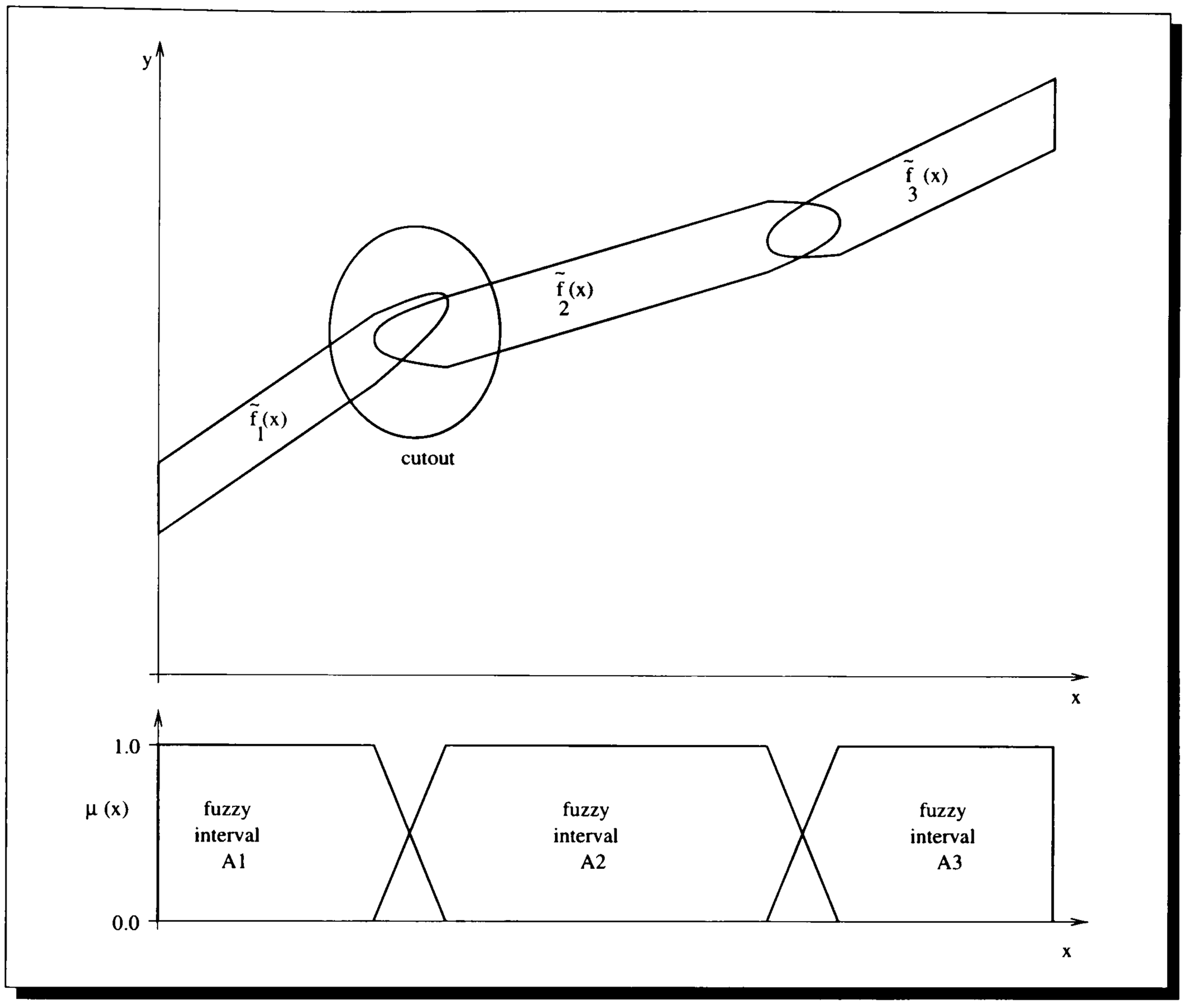


Figure 4.5: Fuzzy Relation Memory

Fig. 4.6 shows a cutout of Fig. 4.5 explaining the transition between two fuzzifying functions at the $\alpha - level = 0.0, 0.5, 1.0$.

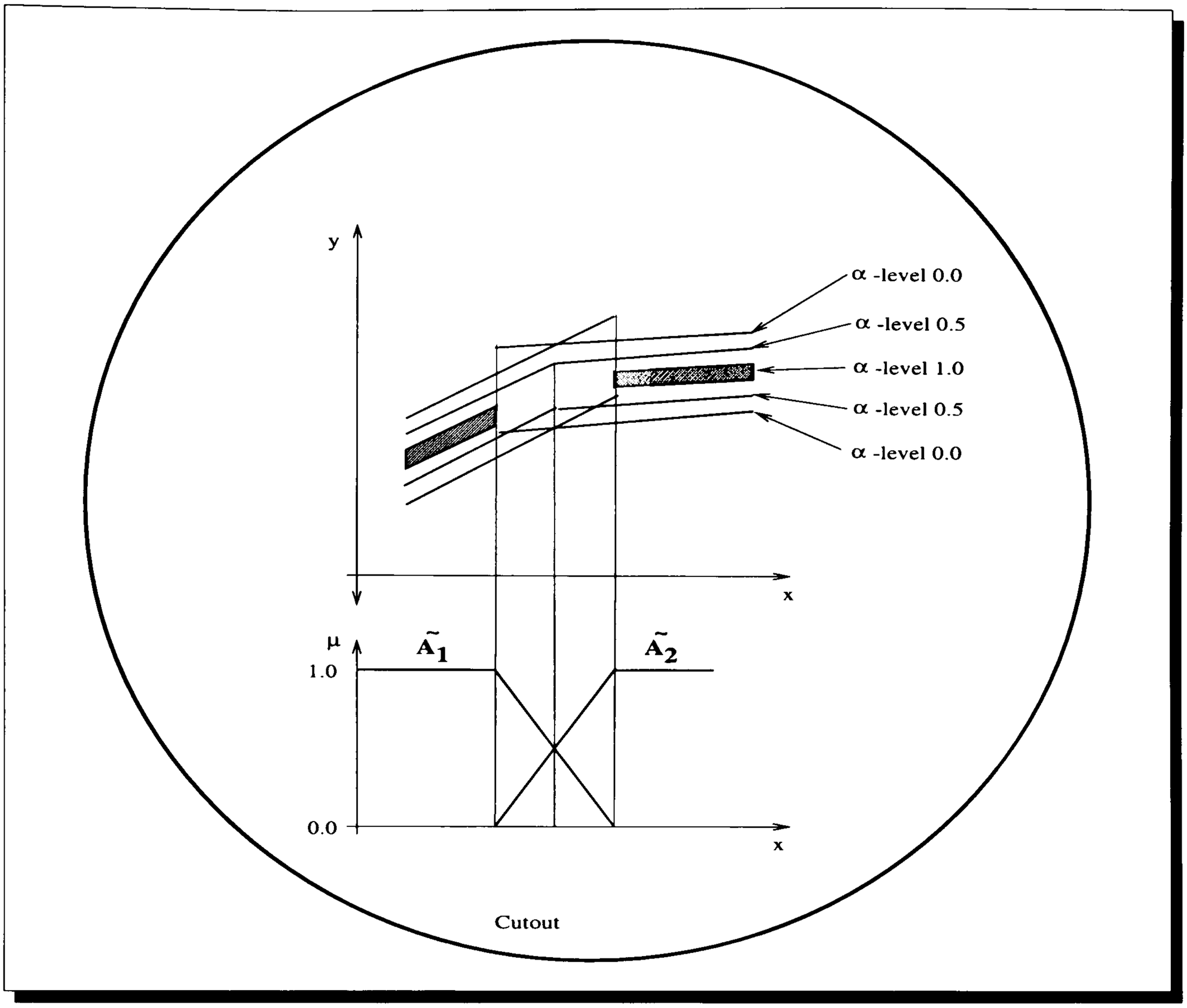


Figure 4.6: Fuzzy Relation Memory

Fuzzy Relation Memories can be used for representing either:

- **Imprecise Signals:** When designing a nonlinear system there are often input signals feeding the model. These dynamic model parameters are not known exactly. Approximating imprecise signals by mathematical functions deceives an exactness which can not be found and can cause a wrong system design (more in Chapter 10.3.3).

- **Imprecise Models:** Modelling very complex nonlinear systems forces approximations. These approximations of the system, for example using linguistic variables, make it necessary to take into account the imprecision involved

in the model (more in Chapter 7). Models build by Fuzzy Relation Memories represent a set of linear models with linear behaviour. Conditions decide which of the linear models is chosen.

Beside the advantage stating the imprecision involved in signal descriptions and system models it is possible to combine these FRMs with ordinary mathematical operators e.g. add, multiply, differentiate, integrate, etc. and compute an overall Fuzzy Relation Memory.

Definition 4.29 (*Fuzzy Relation Memory*): A Fuzzy Relation Memory (FRM) is represented by a canonical rule-based form of fuzzy relational equations

$$\begin{aligned}
\tilde{R}_1&: \quad IF\ x\ is\ \tilde{A}_1,\ THEN\ \tilde{y}_1\ is\ \tilde{f}_1(x)\\
\tilde{R}_2&: \quad IF\ x\ is\ \tilde{A}_2,\ THEN\ \tilde{y}_2\ is\ \tilde{f}_2(x)\\
&\cdot \qquad \cdot\\
&\cdot \qquad \cdot\\
\tilde{R}_n&: \quad IF\ x\ is\ \tilde{A}_n,\ THEN\ \tilde{y}_n\ is\ \tilde{f}_n(x)
\end{aligned}$$

Figure 4.7: Canonical Rule-Based Form Of Fuzzy Relational Equations

whereas $\tilde{A}_n$ are fuzzy sets with a trapezoidal membership function and $\tilde{f}_n(x)$ are fuzzifying functions. Each relation is a Temporal Fuzzifying Function (TFF), therefore, a Fuzzy Relation Memory (FRM) is a set of temporal fuzzifying functions TFFs.

The membership value of the resulting FRM (of all conclusions $\tilde{y}_i$ with $i \in [1, n]$) $\mu_{\tilde{y}_{result}}(x, y)$ has to be computed by considering the membership value μ of every single relation $\tilde{R}_i$ with $i = 1, 2, ..., n$ using the Maximum-Operator

$$\mu_{\tilde{y}_{result}} = max[\mu_{\tilde{y}_1}, \mu_{\tilde{y}_2}, ..., \mu_{\tilde{y}_n}] \tag{4.14}$$

Instead of the Maximum-Operator there could be used other operators, e.g. the Algebraic-Sum-Operator

$$\mu_{\tilde{y}_{result}} = \mu_{\tilde{y}_1} + \mu_{\tilde{y}_2} + ... + \mu_{\tilde{y}_n} - (\mu_{\tilde{y}_1} * \mu_{\tilde{y}_2} * ... * \mu_{\tilde{y}_n}) \tag{4.15}$$

If not explicitly stated, the Maximum-Operator is used in this thesis.

4.4.1 Example: Fuzzy Relation Memory

Fig. 4.8 shows the principle representation of an trapezoidal imprecise voltage signal by Fuzzy Relation Memories.

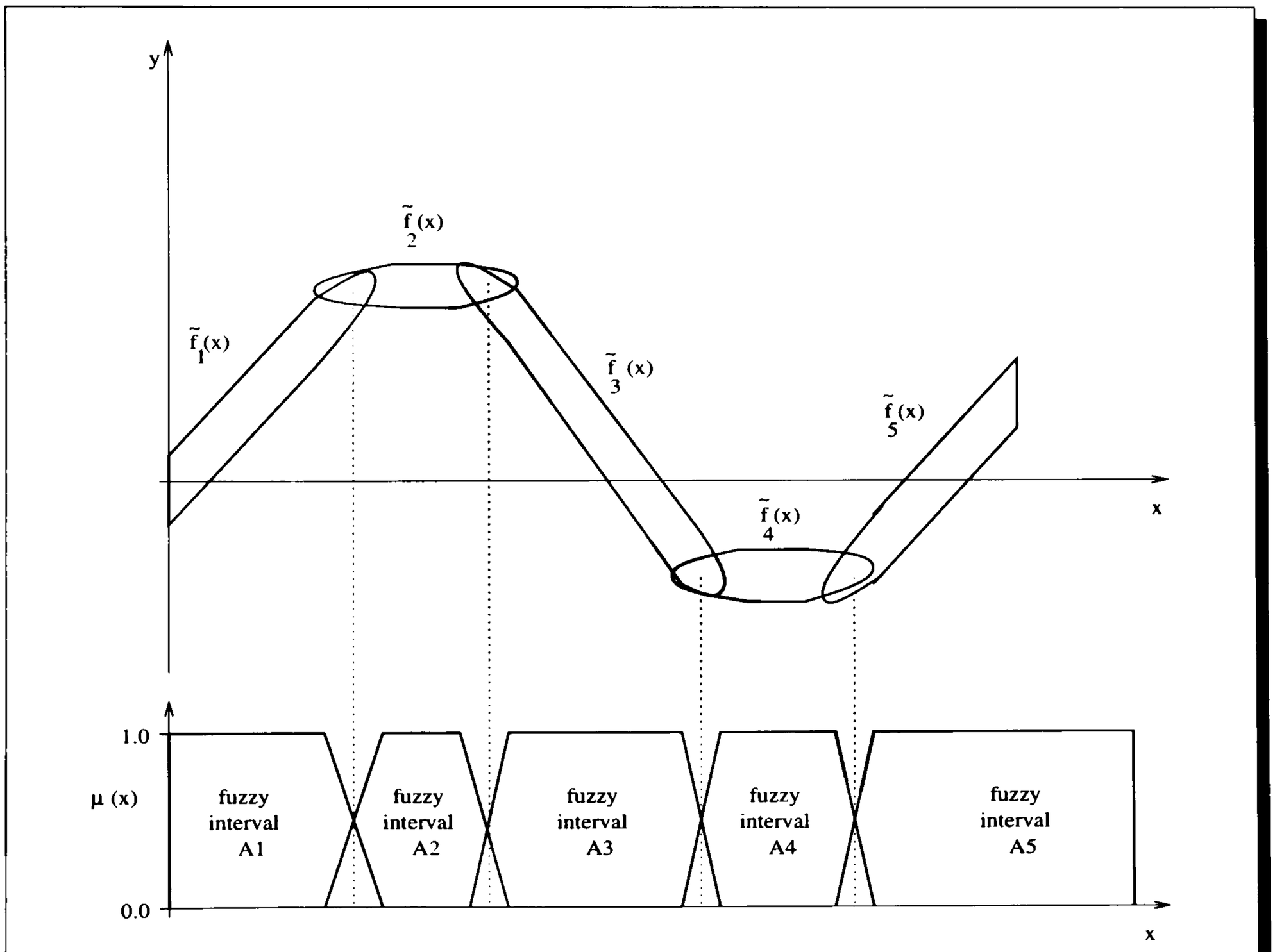


Figure 4.8: Trapezoidal Imprecise Signal by Fuzzy Relation Memory

More specifically the imprecise analogue voltage signal has the following require-

ments:

- an area of the signal should be (certainty factor $\mu = 1.0$)
- an area of the signal is just allowed (certainty factor $\mu > 0.0$)

Fig. 4.9 shows an example with five fuzzy relational equations:

$$\begin{aligned}
\tilde{R}_1&: \quad IF\ x\ is\ \tilde{A}_1,\ THEN\ \tilde{y_1}\ is\ \tilde{f}_1(x)\\
\tilde{R}_2&: \quad IF\ x\ is\ \tilde{A}_2,\ THEN\ \tilde{y_2}\ is\ \tilde{f}_2(x)\\
\tilde{R}_3&: \quad IF\ x\ is\ \tilde{A}_3,\ THEN\ \tilde{y_3}\ is\ \tilde{f}_3(x)\\
\tilde{R}_4&: \quad IF\ x\ is\ \tilde{A}_4,\ THEN\ \tilde{y_4}\ is\ \tilde{f}_4(x)\\
\tilde{R}_5&: \quad IF\ x\ is\ \tilde{A}_5,\ THEN\ \tilde{y_5}\ is\ \tilde{f}_5(x)
\end{aligned}$$

whereas $\tilde{A}_1$, $\tilde{A}_2$, $\tilde{A}_3$, $\tilde{A}_4$, $\tilde{A}_5$, $\tilde{f}_1(x)$, $\tilde{f}_2(x)$, $\tilde{f}_3(x)$, $\tilde{f}_4(x)$, and $\tilde{f}_4(x)$ are defined as follows:

$\tilde{A}_1 = (0.25, 1.75, 0.5, 0.5)$ and linearly defined fuzzifying function $\tilde{f}_1(x)$ with the following α-level functions:

$$f^-_{>0.0}(x) = 2.25\tfrac{V}{s} * x - 1.5V \qquad f^-_{1.0}(x) = 2.25\tfrac{V}{s} * x - 0.5V$$

$$f^+_{1.0}(x) = 2.25\tfrac{V}{s} * x + 0.5V \qquad f^+_{>0.0}(x) = 2.25\tfrac{V}{s} * x + 1.5V$$

$\tilde{A}_2 = (2.25, 3.75, 0.5, 0.5)$ and linearly defined fuzzifying function $\tilde{f}_2(x)$ with the following α-level functions:

$$f^-_{>0.0}(x) = 3V \qquad f^-_{1.0}(x) = 4V$$

$$f^+_{1.0}(x) = 5V \qquad f^+_{>0.0}(x) = 6V$$

$\tilde{A}_3 = (4.25, 7.75, 0.5, 0.5)$ and linearly defined fuzzifying function $\tilde{f}_3(x)$ with the following α-level functions:

$$f^-_{>0.0}(x) = -2.25\frac{V}{s} * x + 12V \qquad f^-_{1.0}(x) = -2.25\frac{V}{s} * x + 13V$$

$$f^+_{1.0}(x) = -2.25\frac{V}{s} * x + 14V \qquad f^+_{>0.0}(x) = -2.25\frac{V}{s} * x + 15V$$

$\tilde{A}_4 = (8.25, 9.75, 0.5, 0.5)$ and linearly defined fuzzifying function $\tilde{f}_4(x)$ with the following α-level functions:

$$f^-_{>0.0}(x) = -6V \qquad f^-_{1.0}(x) = -5V$$

$$f^+_{1.0}(x) = -4V \qquad f^+_{>0.0}(x) = -3V$$

$\tilde{A}_5 = (10.25, 11.75, 0.5, 0.5)$ and linearly defined fuzzifying function $\tilde{f}_5(x)$ with the following α-level functions:

$$f^-_{>0.0}(x) = 2.25\frac{V}{s} * x - 28.5V \qquad f^-_{1.0}(x) = 2.25\frac{V}{s} * x - 27.5V$$

$$f^+_{1.0}(x) = 2.25\frac{V}{s} * x - 26.5V \qquad f^+_{>0.0}(x) = 2.25\frac{V}{s} * x - 25.5V$$

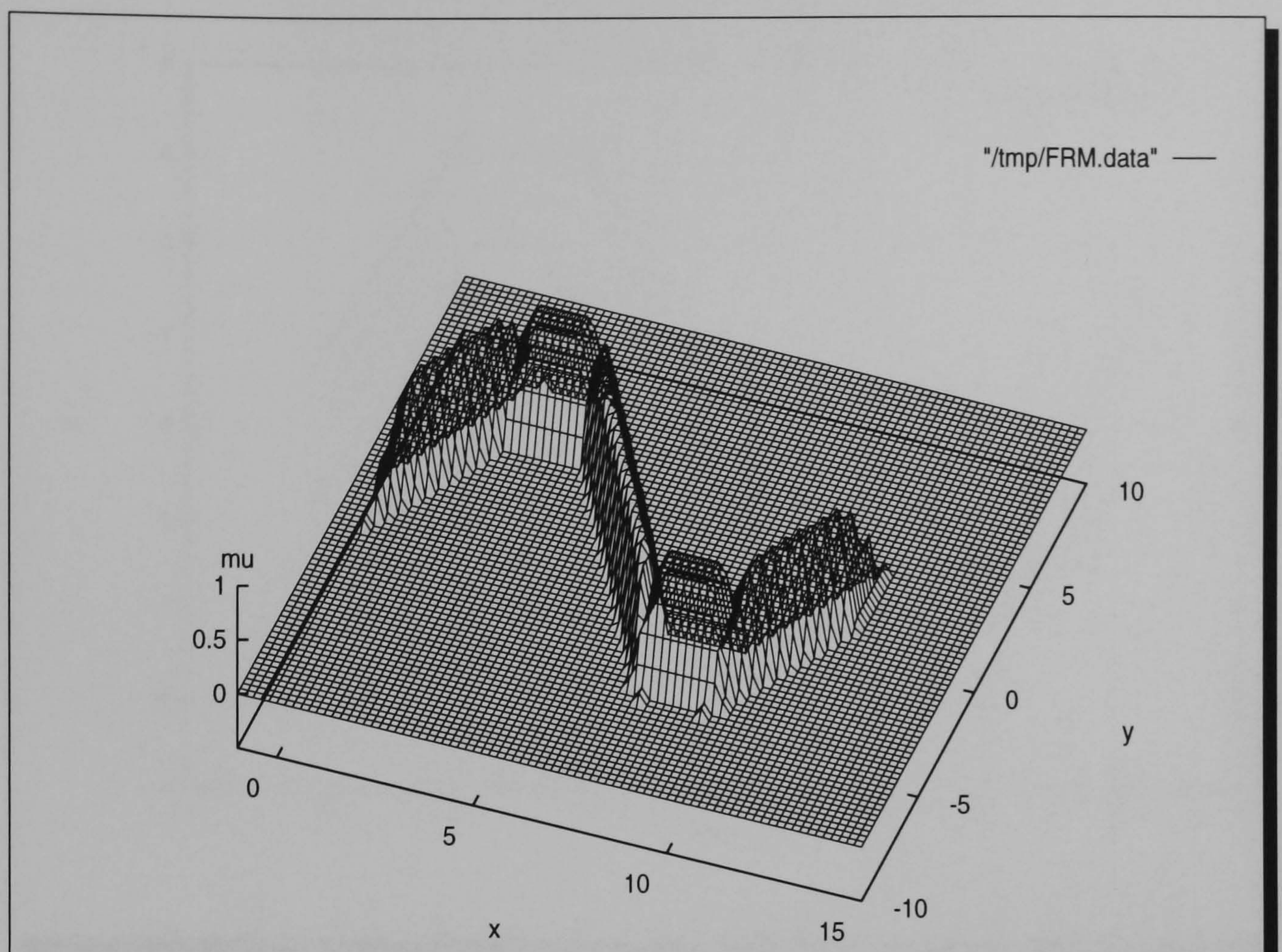


Figure 4.9: 3D-View Fuzzy Relation Memory

Fig. 4.10 shows a 2-dimensional view of the imprecise signal example. The α-level, $\mu = 0$ and $\mu = 1.0$, are displayed.

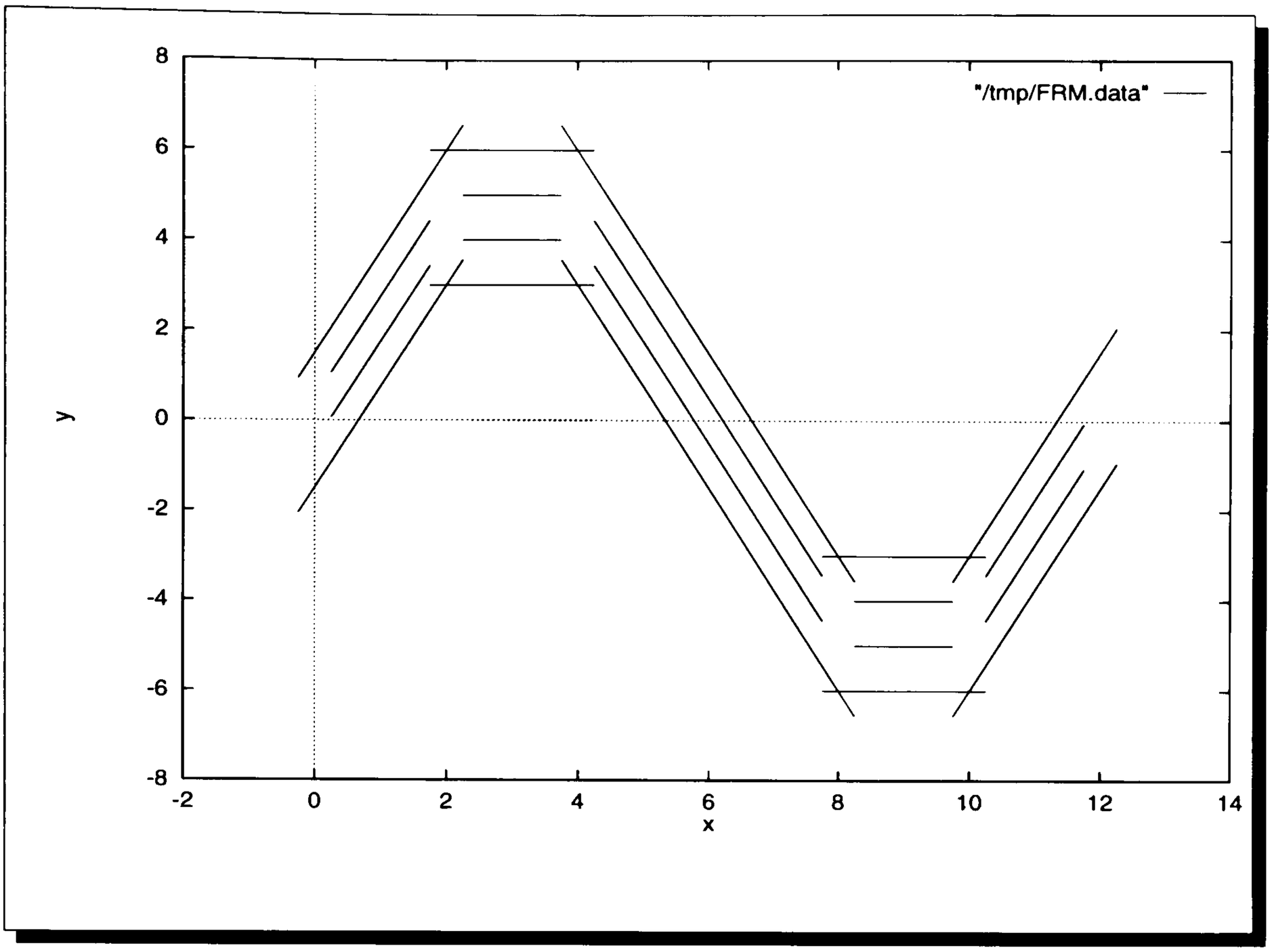


Figure 4.10: 2D-View Fuzzy Relation Memory

4.5 Qualitative Signals Represented by Fuzzy Relation Memories

As stated in [Lunze, 1995] signals could be represented by dynamic confluences. Dynamic confluences are qualitative signals that are a totally time-ordered set of landmark values. Dynamic confluences do not tell at which particular time point a qualitative value is changed. Only the order of changes is known. Qualitative reasoning maps the continuous phenomenon time to the discrete form of a sequence of well distinguished points and ranges on the time axes:

$$T = \{[t_i, t_i] | t_i \in K\} \cup \{(t_i, t_{i+1}) | t_i, t_{i+1} \in K, t_i < t_{i+1}\} \tag{4.16}$$

where K is a finite set of numbers.

A common qualitative representation of an analogue signal in the qualitative reasoning community is, for example, $y(t) = cos(t)\ with\ t = [0, \frac{5}{2}\Pi]$:

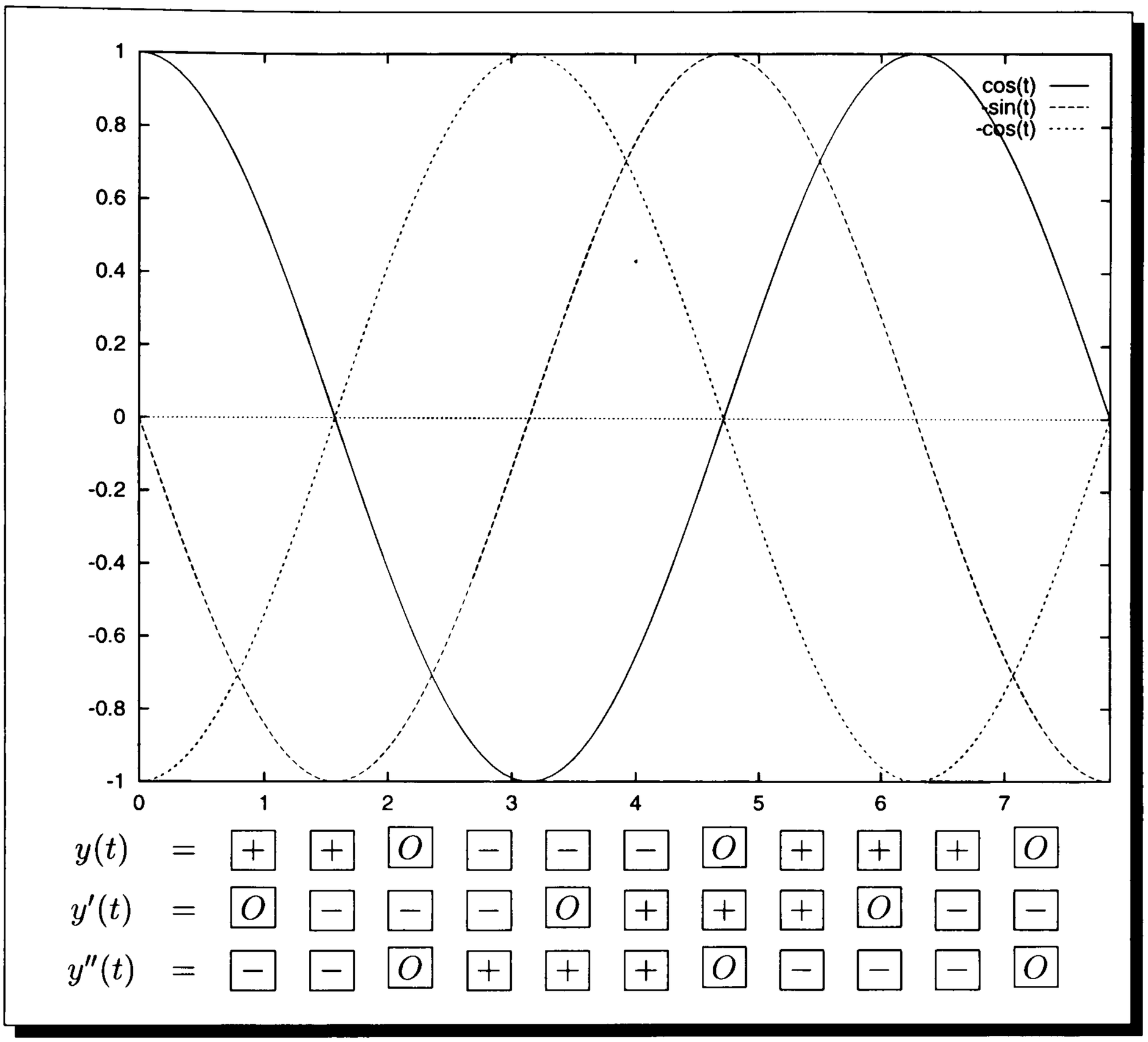


Figure 4.11: Qualitative Signal

By using this kind of representation the ordering problem occurs. Two parameters move towards the same landmark. Which one reaches the landmark first? This is called the *ordering problem.* When it is not decidable, many different behaviours occur. Further it is not possible to propagate the duration time of a state or the effect of an influence over a longer time. The ordered landmarks are associated with time fuzzy intervals which solves the ordering problem.

4.5.1 Example: Qualitative Signal Represented by Fuzzy Relation Memory

Suppose the qualitative signal shown in Fig. 4.11 is represented by a Fuzzy Relation Memory.

The qualitative fuzzy quantity $\mathbb{Q}$-domain is given by the three qualitative values $\mathbb{Q} = \{n\tilde{e}g, n\tilde{u}ll, p\tilde{o}s\}$ which are defined by the fuzzy intervals and a fuzzy number

qualitative $n\tilde{e}g$ is represented by the fuzzy interval: $(-10, -1, 1, 1)$

qualitative $n\tilde{u}ll$ is represented by the fuzzy number: $(0, 1, 1)$

qualitative $p\tilde{o}s$ is represented by the fuzzy interval: $(1, 10, 1, 1)$

and the time fuzzy intervals

$$\begin{aligned}
\tilde{T}_1 &= (0, \tfrac{\Pi}{2}, \tfrac{\Pi}{10}, \tfrac{\Pi}{10}) \\
\tilde{T}_2 &= (\tfrac{\Pi}{2}, \Pi, \tfrac{\Pi}{10}, \tfrac{\Pi}{10}) \\
\tilde{T}_3 &= (\Pi, \tfrac{3*\Pi}{2}, \tfrac{\Pi}{10}, \tfrac{\Pi}{10}) \\
\tilde{T}_4 &= (\tfrac{3*\Pi}{2}, 2*\Pi, \tfrac{\Pi}{10}, \tfrac{\Pi}{10}) \\
\tilde{T}_5 &= (2*\Pi, \tfrac{5*\Pi}{2}, \tfrac{\Pi}{10}, \tfrac{\Pi}{10})
\end{aligned}$$

This results in the following fuzzy relational equations:

$$\begin{aligned}
\tilde{R}_1\text{:} &\quad IF\ t\ is\ \tilde{T}_1,\ THEN\ \tilde{y}_1\ is\ \tilde{f}_1(t) \\
\tilde{R}_2\text{:} &\quad IF\ t\ is\ \tilde{T}_2,\ THEN\ \tilde{y}_2\ is\ \tilde{f}_2(t) \\
\tilde{R}_3\text{:} &\quad IF\ t\ is\ \tilde{T}_3,\ THEN\ \tilde{y}_3\ is\ \tilde{f}_3(t) \\
\tilde{R}_4\text{:} &\quad IF\ t\ is\ \tilde{T}_4,\ THEN\ \tilde{y}_4\ is\ \tilde{f}_4(t) \\
\tilde{R}_5\text{:} &\quad IF\ t\ is\ \tilde{T}_5,\ THEN\ \tilde{y}_5\ is\ \tilde{f}_5(t)
\end{aligned}$$

whereas $\tilde{f}_1(t)$, $\tilde{f}_2(t)$, $\tilde{f}_3(t)$, $\tilde{f}_4(t)$, and $\tilde{f}_5(t)$ are defined as follows:

$\tilde{f}_1(t)$ valid at $\tilde{T}_1$ is $p\tilde{o}s$ and defined by the following α-level functions:

$$\begin{aligned}
f^-_{>0.0}(t) &= 0 \qquad f^-_{1.0}(t) = 1 \\
f^+_{1.0}(t) &= 10 \qquad f^+_{>0.0}(t) = 11
\end{aligned}$$

$\tilde{f}_2(t)$ valid at $\tilde{T}_2$ is $n\tilde{e}g$ and defined by the following α-level functions:

$$f^-_{>0.0}(t) = -11 \quad f^-_{1.0}(t) = -10$$
$$f^+_{1.0}(t) = -1 \quad f^+_{>0.0}(t) = 0$$

$\tilde{f}_3(t)$ valid at $\tilde{T}_3$ is $n\tilde{e}g$ and defined by the following α-level functions:

$$f^-_{>0.0}(t) = -11 \quad f^-_{1.0}(t) = -10$$
$$f^+_{1.0}(t) = -1 \quad f^+_{>0.0}(t) = 0$$

$\tilde{f}_4(t)$ valid at $\tilde{T}_4$ is $p\tilde{o}s$ and defined by the following α-level functions:

$$f^-_{>0.0}(t) = 0 \quad f^-_{1.0}(t) = 1$$
$$f^+_{1.0}(t) = 10 \quad f^+_{>0.0}(t) = 11$$

$\tilde{f}_5(t)$ valid at $\tilde{T}_5$ is $p\tilde{o}s$ and defined by the following α-level functions:

$$f^-_{>0.0}(t) = 0 \quad f^-_{1.0}(t) = 1$$
$$f^+_{1.0}(t) = 10 \quad f^+_{>0.0}(t) = 11$$

Fig. 4.12 visualizes the Fuzzy Relation Memory defined above.

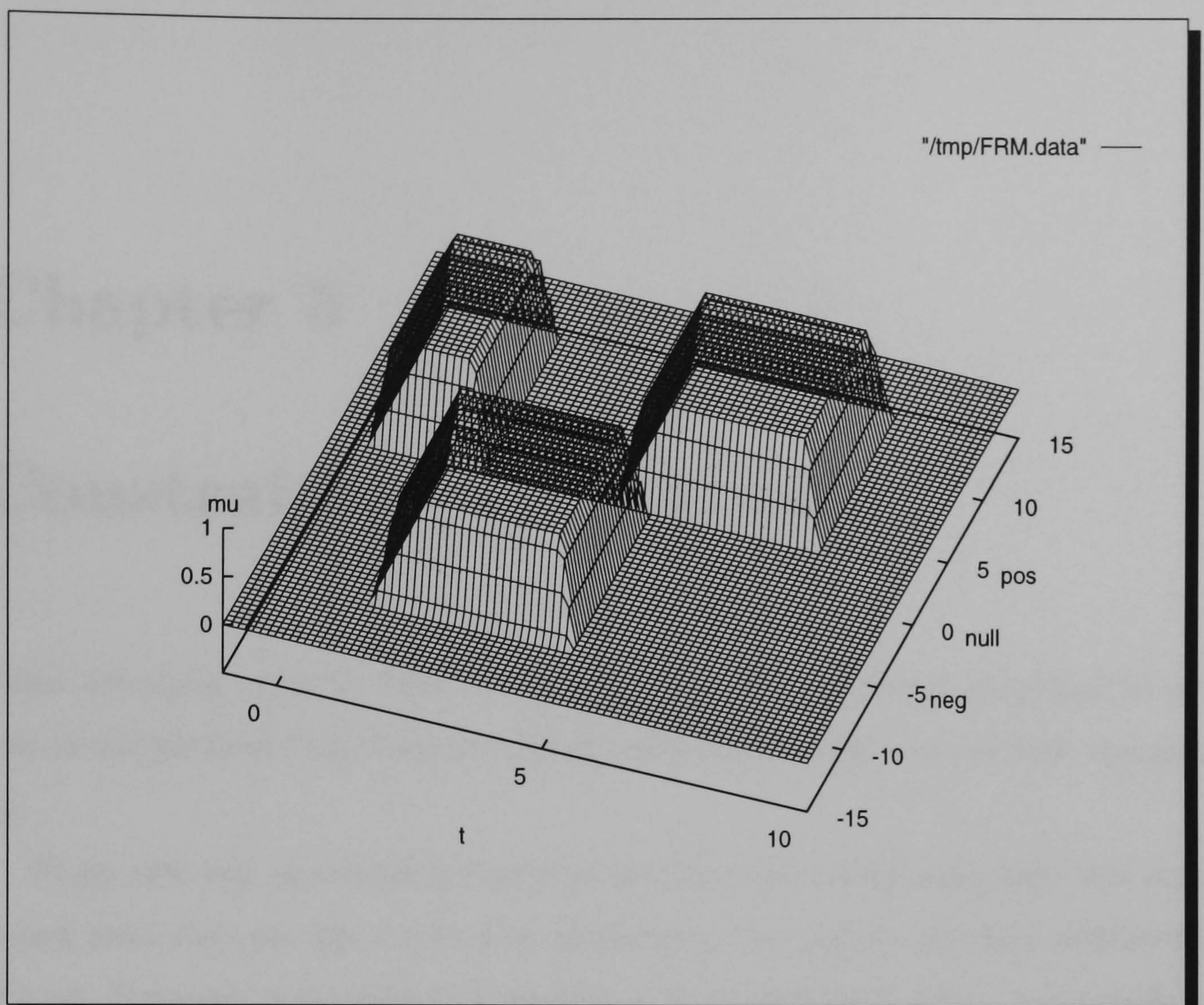


Figure 4.12: Qualitative Signal Represented by FRM

4.6 Conclusion

Defining dynamic model parameters of nonlinear systems not known precisely using Fuzzy Relational Memories (FRMs) has advantages: It enables the designer to specify imprecise input signals, imprecise output signals, or imprecise dynamic parameters of the nonlinear system model as humans are used to. In Chapter 10 parts of a user interface of a drawing tool is shown that helps design engineers to specify imprecise signals directly with a user interface of a prototype application.

Fuzzy Relational Memories can represent qualitative signals which solve the ordering problem of qualitative simulation.

Chapter 5

Constraints — Relations

There are many different kind of relations. Relations represent mappings for sets just as mathematical functions with their mathematical operators or logic operators do.

Fuzzy sets and membership functions are the reason why fuzzy logic was introduced; since they provide a means of representing the concept of vague membership of a set. However, this ability to map data to fuzzy set memberships is not useful in itself as we also require a set of operators for combining this information and making inferences about its state. Fuzzy logical operators as AND, OR, NOT, etc. are important but out of the scope of this work. More relevant are the classical mathematical operations to do reasoning using mathematical models or combine basic models to more complex models. The fundamental concept to extend the classical operations to fuzzy arithmetic and fuzzy algebraic operations is accomplished with Zadeh's extension principle [Zadeh, 1975a]. It provides a general method for extending standard mathematical concepts in order to deal with fuzzy quantities. The common mathematical operators for fuzzy numbers and fuzzy intervals are defined in Appendix A *Fuzzy Reasoning Basics* as described in [Kaufmann and Gupta, 1991]. The operators equality, similarity, addition, subtraction, multiplication, division, derivation, and integration for the new introduced fuzzy type Fuzzy Curves — Fuzzy

Relation Memories are discussed in detail in the following sections.

5.1 Vertex Method

The membership functions of the variables are made discrete for computational convenience which has a serious disadvantage. Using Zadeh's extension principle [Zadeh, 1975a] with discretized membership functions irregular and erroneous membership functions of the output variables are determined. This problem does not arise because of any inherent problem in the extension principle itself. It arises when continuous-valued functions are made discrete, then allowed to propagate from the input domain to the output domain using the extension principle. For example, two fuzzy sets $\tilde{X}$ and $\tilde{Y}$ with the membership functions as shown in Fig. 5.1.

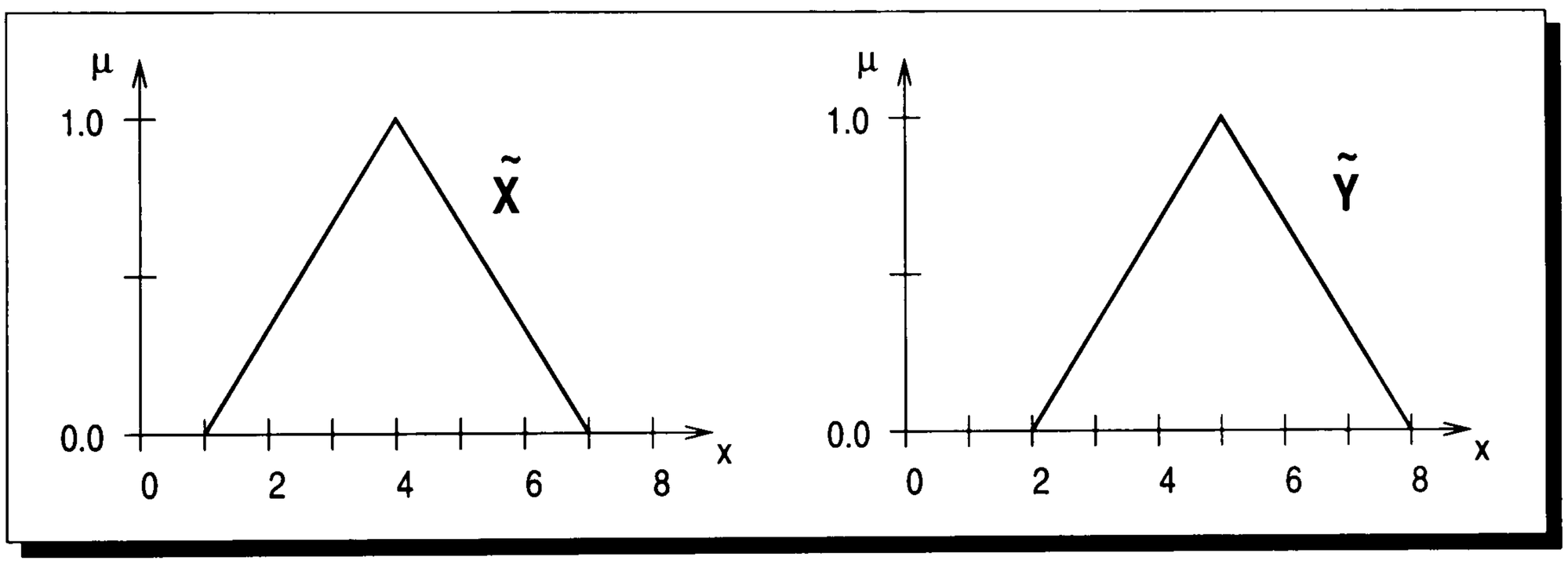


Figure 5.1: Fuzzy Sets $\tilde{X}$ and $\tilde{Y}$

The two fuzzy sets are discretized at seven points:

$$\tilde{X} = \{(1,0); (2,0.33); (3,0.66); (4,1.0); (5,0.66); (6,0.33); (7,0)\}$$

$$\tilde{Y} = \{(2,0); (3,0.33); (4,0.66); (5,1.0); (6,0.66); (7,0.33); (8,0)\}$$

Suppose $\tilde{X} * \tilde{Y}$ has to be computed. Using Zadeh's extension principle, first the Cartesian Product has to be computed:

$$\begin{aligned}\tilde{X} \times \tilde{Y} = \quad & \{(2, 0.0); (3, 0.0); (4, 0.0); (5, 0.0); (6, 0.33); (7, 0.0); (8, 0.33); (9, 0.33); \\ & (10, 0.33); (12, 0.66); (14, 0.33); (15, 0.66); (16, 0.33); (18, 0.66); (20, 1.0); \\ & (21, 0.33); (24, 0.66); (25, 0.66); (28, 0.33); (30, 0.66); (32, 0.0); (35, 0.33); \\ & (36, 0.33); (40, 0.0); (42, 0.33); (48, 0.0); (49, 0.0); (56, 0.0)\}\end{aligned}$$

The result of the operation $\tilde{X} * \tilde{Y}$ for a discretization level of seven points is plotted in Fig. 5.2:

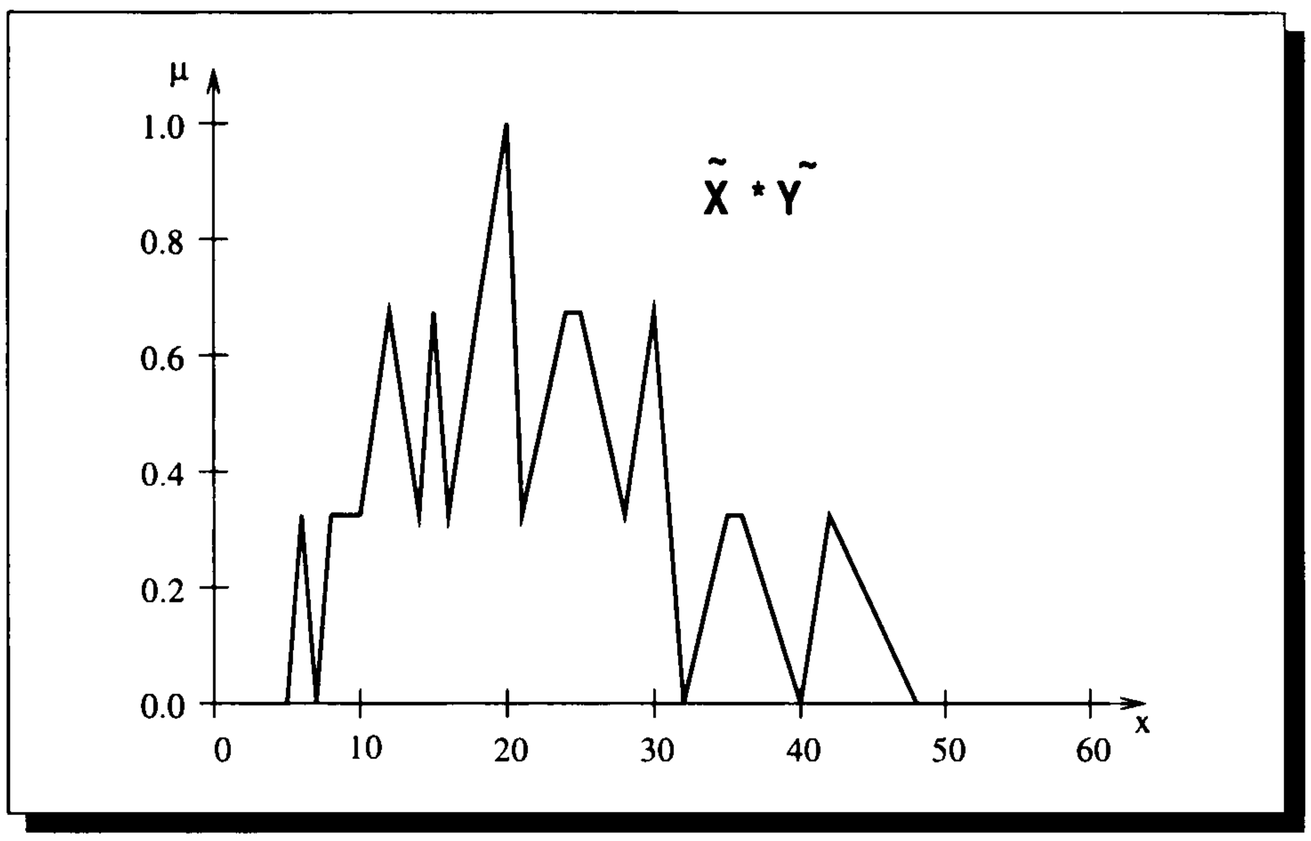


Figure 5.2: $\tilde{X} * \tilde{Y}$ with 7 point discretization of $\tilde{X}$ and $\tilde{Y}$

There are methods proposed for implementing the extension principle for continuous-valued functions and mappings overcome the drawback described above, e.g. *Vertex Method* [Dong and Shah, 1987], *DSW Algorithm* [Dong et al., 1985], and *Restricted DSW Algorithm* [Givens and Tahani, 1987]. In this thesis the *Vertex Method* is used because it is a simple method and implemented easily.

The *Vertex Method* [Dong and Shah, 1987] is based on a combination of the α-cut concept and standard interval analysis. The vertex method can prevent abnormality in the output membership function due to application of the discretization technique on the fuzzy variables' domain, and it can prevent the widening of the resulting function value set due the multiple occurrences of variables in the functional expression by conventional interval analysis methods.

Definition 5.30 (*Vertex Method*): Suppose a functional mapping is given by n input, i.e., $y = f(x_1, x_2, ..., x_n)$, then the input space can be represented by an n-dimensional Cartesian region. Each of the input variables can be described by an interval, say $I_{i\alpha}$ at a specific α-cut, where

$$I_{i\alpha} = [a_i, b_i] \tag{5.1}$$

When the mapping $y = f(x_1, x_2, ..., x_n)$ is continuous in the n-dimensional Cartesian region and when also there is no extreme point[1] (no maximum or minimum) in this region, the value of the interval function for a particular α-cut can be obtained by

$$\begin{aligned} B_\alpha &= f(I_{1\alpha}, I_{2\alpha}, ..., I_{n\alpha}) \\ &= [\underset{i}{min}(f(c_i)), \underset{i}{max}(f(c_i))] \; with \; i = 1, 2, ..., N \end{aligned} \tag{5.2}$$

where c_i is the coordinate of the ith vertex representing the n-dimensional Cartesian region.

Using the example described above with the two fuzzy sets $\tilde{X}$ and $\tilde{Y}$ shown in Fig. 5.1 discretized with 7 points the new membership function of the product is determined by the *Vertex Method* as follows:

$I_{>0.0}$: Support for $\tilde{X}$ is the interval $[1, 7]$ and support for $\tilde{Y}$ is the interval $[2, 8]$.

$$\begin{aligned} min[1*2, 1*8, 7*2, 7*8] = min[2, 8, 14, 56] &= 2 \\ max[1*2, 1*8, 7*2, 7*8] = max[2, 8, 14, 56] &= 56 \\ B_{>0.0} &= [2, 56] \end{aligned}$$

$I_{0.33}$: Support for $\tilde{X}$ is the interval $[2, 6]$ and support for $\tilde{Y}$ is the interval $[3, 7]$.

$$\begin{aligned} min[2*3, 2*7, 6*3, 6*7] = min[6, 14, 18, 42] &= 6 \\ max[2*3, 2*7, 6*3, 6*7] = max[6, 14, 18, 42] &= 42 \\ B_{0.33} &= [6, 42] \end{aligned}$$

[1] Known extreme points are added as additional vertices.

$I_{0.66}$: Support for $\tilde{X}$ is the interval $[3, 5]$ and support for $\tilde{Y}$ is the interval $[4, 6]$.

$$\begin{aligned} min[3*4, 3*6, 5*4, 5*6] = min[12, 18, 20, 30] &= 12 \\ max[3*4, 3*6, 5*4, 5*6] = max[12, 18, 20, 30] &= 30 \\ B_{0.66} &= [12, 30] \end{aligned}$$

$I_{1.0}$: Support for $\tilde{X}$ is the interval $[4, 4]$ and support for $\tilde{Y}$ is the interval $[5, 5]$.

$$\begin{aligned} min[4*5, 4*5, 4*5, 4*5] = min[20, 20, 20, 20] &= 20 \\ max[4*5, 4*5, 4*5, 4*5] = max[20, 20, 20, 20] &= 20 \\ B_{1.0} &= [20, 20] \end{aligned}$$

The result of the plotting the four α-cut levels is shown in Fig. 5.3:

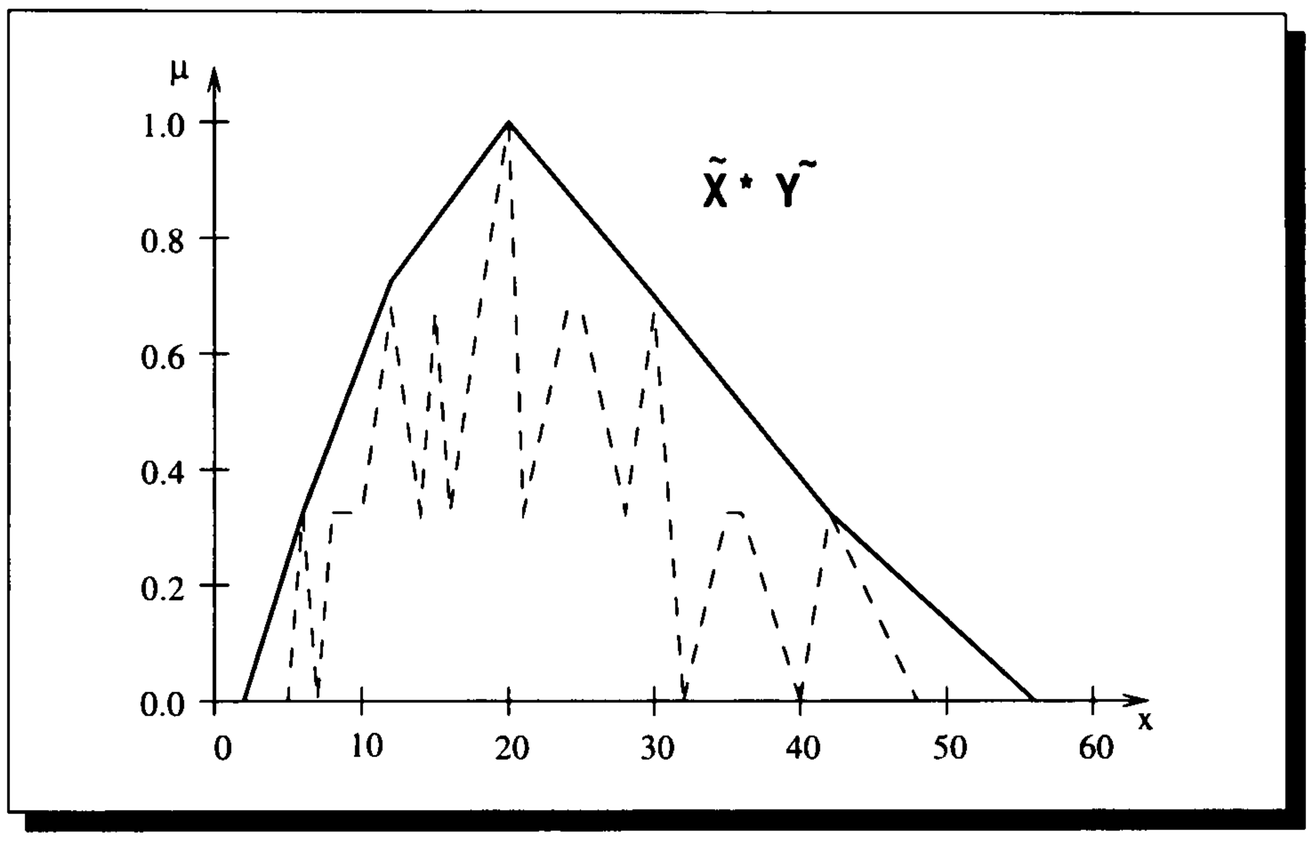


Figure 5.3: Vertex Method: $\tilde{X} * \tilde{Y}$ with 7 point discretization of $\tilde{X}$ and $\tilde{Y}$

5.2 Equality of Fuzzy Curves

Two fuzzy numbers $\tilde{A}$ and $\tilde{B}$ are equal, when their membership functions $\mu_A(x) = \mu_B(x)$ are equal. This was Zadeh's definition of equality [Zadeh, 1972]. This is also true for Fuzzy Curves but it needs a little more effort for proving.

Definition 5.31 (*Equality of Fuzzy Curves*): Let FC_1 and FC_2 be two Fuzzy Curves and $TFF_1(i)$ and $TFF_2(i)$ be their Temporal Fuzzifying Functions.

The equality of two synchronous (defined in Section 5.4) Fuzzy Curves is proved, if and only if, all their Temporal Fuzzifying Functions are equal, so we can then write

$$FC_1 = FC_2 \equiv [TFF_1(i) = TFF_2(i)] \; with \; i = 1, 2, ..., n \tag{5.3}$$

Every Temporal Fuzzifying Function of the Fuzzy Curve FC_1 has to be equal the Temporal Fuzzifying Functions of the Fuzzy Curve FC_2. The equality of two Fuzzy Curves can only be proved if they are synchronized. They are synchronized if the two membership functions of the time Fuzzy Intervals, $\tilde{I}_1$ and $\tilde{I}_2$ are equal. This means that it is enough to check whether the fuzzifying functions of the fuzzy relation memories are equal or not.

Definition 5.32 (*Equality of Temporal Fuzzifying Functions of Fuzzy Relation Memories*): The two Temporal Fuzzifying Functions TFF_1 and TFF_2 of the synchronized Fuzzy Relation Memories (see Section 5.4) are equal, if and only if, the two membership functions of the Fuzzifying Functions are equal.

Suppose FC_1 is defined

$$FC_1 : IF \; x \; is \; \tilde{I}_1, \; THEN \; \tilde{y} \; is \; \tilde{f}_1(x)$$

and FC_2 is defined

$$FC_2 : IF \; x \; is \; \tilde{I}_2, \; THEN \; \tilde{y} \; is \; \tilde{f}_2(x)$$

with $\tilde{I}_1 = \tilde{I}_2$ so it can be written

$$\mu_{\tilde{f}_1(x)} = \mu_{\tilde{f}_2(x)} \tag{5.4}$$

Definition 5.33 (*Inequality of Fuzzy Curves*): Let FC_1 and FC_2 be two Fuzzy Curves and $TFF_1(i)$ and $TFF_2(i)$ be their Temporal Fuzzifying Functions. The inequality of two synchronous Fuzzy Curves is proved, if the Fuzzy Curves can not be proved equal, as defined in Section 5.2.

5.3 Similarity Measurement between Fuzzy Curves

The notion of similarity is essentially a generalization of the notion of equality and well discussed for fuzzy values in Zadeh's paper: "Similarity Relations and Fuzzy Orderings" [Zadeh, 1977]. There are quite a number of fuzzy similarity measurements defined in the literature [Dubois and Prade, 1980], [Zadeh, 1971]. The work "Fuzzy Similarity" [Reich, 1997a] compares various similarity measurements using a simple voltage divider circuit. Most suitable seems to be the "Relative Hamming distance" [Kaufmann, 1975] to compare two Fuzzy Curves. The Relative Hamming distance is defined for fuzzy sets in general.

Definition 5.34 (*Relative Hamming Distance*): If there is no difference between the two fuzzy sets then similarity measurement is 1.

$$S(A,B) = 1 - \text{difference}(A,B) \tag{5.5}$$

$$S(A,B) = 1 - \parallel A \bigtriangledown B \parallel$$

with the Relative Hamming distance [Kaufmann, 1975] defined as:

$$\parallel A \bigtriangledown B \parallel \tag{5.6}$$

The relative cardinality is defined as:

$$\parallel A \bigtriangledown B \parallel = \frac{|A \bigtriangledown B|}{|X|} \tag{5.7}$$

with the scalar cardinality defined as:

$$|A \bigtriangledown B| = \sum_{x \in X} \mu_{A \bigtriangledown B}(x) \tag{5.8}$$

with:

$$\mu_{A \bigtriangledown B}(x) = \mid \mu_A(x) - \mu_B(x) \mid \tag{5.9}$$

The Relative Hamming distance has to be applied to each single α-level and to each linear function of the fuzzy curve.

Definition 5.35 (*Similarity Measurement of Fuzzy Curves*): Let FC1 and FC2 be two Fuzzy Curves. Then their similarity is:

$$S_{fuzzy\ curve} = \sum S_{temporal\ fuzzifying\ function}(i) \qquad \text{i=1,2,...,n} \qquad (5.10)$$

Let TFF1 and TFF2 be two Temporal Fuzzifying Functions. Then their similarity is defined as:

$$S_{temporal\ fuzzifying\ function}(TFF1, TFF2)_i = 1 - \parallel TFF1 \bigtriangledown TFF2 \parallel \qquad (5.11)$$

$$\parallel TFF1 \bigtriangledown TFF2 \parallel = \frac{|TFF1 \bigtriangledown TFF2|}{|X|} \qquad (5.12)$$

$$|TFF1 \bigtriangledown TFF2| = \sum_{x \in X} \mu_{TFF1 \bigtriangledown TFF2}(x) \qquad (5.13)$$

$$\mu_{TFF1 \bigtriangledown TFF2} = \mid \mu_{fuzzifying\ function\ 1}(x)_i - \mu_{fuzzifying\ function\ 2}(x)_i \mid \qquad (5.14)$$

with i=1,2,...,n.

5.4 Synchronization of Fuzzy Curves

Definition 5.36 (*Synchronization of Fuzzy Curves*): The synchronization of two Fuzzy Curves FC_1 and FC_2 whose Temporal Fuzzy Intervals are the same consist in finding two Fuzzy Curves FC_1' and FC_2' such that:

- FC_1 and FC_1' are equal
- FC_2 and FC_2' are equal
- FC_1' and FC_2' have the same number N of Temporal Fuzzifying Functions $TFF(i)$ with $i = 1, 2, ..., N$

- $\tilde{A}'(i)$ with $i = 1, 2, ..., N$ is the Fuzzy Interval input of a Temporal Fuzzifying Function

Then $\forall i \in [1, n]$ the Fuzzy Interval input $\tilde{A_1}'(i) = \tilde{A_2}'(i)$.

Fig. 5.4 shows an example of synchronization of two Fuzzy Curves. Only the Fuzzy Intervals, the preconditions of the IF-THEN rules, are displayed in Fig. 5.4.

FC_1:

$$\tilde{R}_{11}: \quad IF\ x\ is\ \tilde{A}_{11},\ THEN\ \tilde{y_1}\ is\ \tilde{f}_{11}(x)$$
$$\tilde{R}_{12}: \quad IF\ x\ is\ \tilde{A}_{12},\ THEN\ \tilde{y_2}\ is\ \tilde{f}_{12}(x)$$

and FC_2:

$$\tilde{R}_{21}: \quad IF\ x\ is\ \tilde{A}_{21},\ THEN\ \tilde{y_1}\ is\ \tilde{f}_{21}(x)$$
$$\tilde{R}_{22}: \quad IF\ x\ is\ \tilde{A}_{22},\ THEN\ \tilde{y_2}\ is\ \tilde{f}_{22}(x)$$
$$\tilde{R}_{23}: \quad IF\ x\ is\ \tilde{A}_{23},\ THEN\ \tilde{y_3}\ is\ \tilde{f}_{23}(x)$$

Synchronizing the two fuzzy curves results in:

FC'_1:

$$\tilde{R}'_{11}: \quad IF\ x\ is\ \tilde{A}'_{11},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{11}(x)$$
$$\tilde{R}'_{12}: \quad IF\ x\ is\ \tilde{A}'_{12},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{12}(x)$$
$$\tilde{R}'_{13}: \quad IF\ x\ is\ \tilde{A}'_{13},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{13}(x)$$
$$\tilde{R}'_{14}: \quad IF\ x\ is\ \tilde{A}'_{14},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{14}(x)$$

with

$$\tilde{f}'_{11}: \quad = \tilde{f}_{11}(x) \text{ and } \tilde{A}'_{11} = \tilde{A}_{11} \cap \tilde{A}_{21}$$
$$\tilde{f}'_{12}: \quad = \tilde{f}_{11}(x) \text{ and } \tilde{A}'_{12} = \tilde{A}_{11} \cap \tilde{A}_{22}$$
$$\tilde{f}'_{13}: \quad = \tilde{f}_{12}(x) \text{ and } \tilde{A}'_{13} = \tilde{A}_{12} \cap \tilde{A}_{22}$$
$$\tilde{f}'_{14}: \quad = \tilde{f}_{12}(x) \text{ and } \tilde{A}'_{14} = \tilde{A}_{12} \cap \tilde{A}_{23}$$

FC'_2:

$$\begin{aligned}
\tilde{R}'_{21}\text{:} &\quad IF\ x\ is\ \tilde{A}'_{21},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{21}(x)\\
\tilde{R}'_{22}\text{:} &\quad IF\ x\ is\ \tilde{A}'_{22},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{22}(x)\\
\tilde{R}'_{23}\text{:} &\quad IF\ x\ is\ \tilde{A}'_{23},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{23}(x)\\
\tilde{R}'_{24}\text{:} &\quad IF\ x\ is\ \tilde{A}'_{24},\ THEN\ \tilde{y}'\ is\ \tilde{f}'_{24}(x)
\end{aligned}$$

with

$$\begin{aligned}
\tilde{f}'_{21}\text{:} &\quad = \tilde{f}_{21}(x) \text{ and } \tilde{A}'_{21} = \tilde{A}_{11} \cap \tilde{A}_{21}\\
\tilde{f}'_{22}\text{:} &\quad = \tilde{f}_{22}(x) \text{ and } \tilde{A}'_{22} = \tilde{A}_{11} \cap \tilde{A}_{22}\\
\tilde{f}'_{23}\text{:} &\quad = \tilde{f}_{22}(x) \text{ and } \tilde{A}'_{23} = \tilde{A}_{12} \cap \tilde{A}_{22}\\
\tilde{f}'_{24}\text{:} &\quad = \tilde{f}_{23}(x) \text{ and } \tilde{A}'_{24} = \tilde{A}_{12} \cap \tilde{A}_{23}
\end{aligned}$$

The minimal synchronization of two Fuzzy Curves whose Temporal Fuzzy Intervals are the same always exists and is unique.

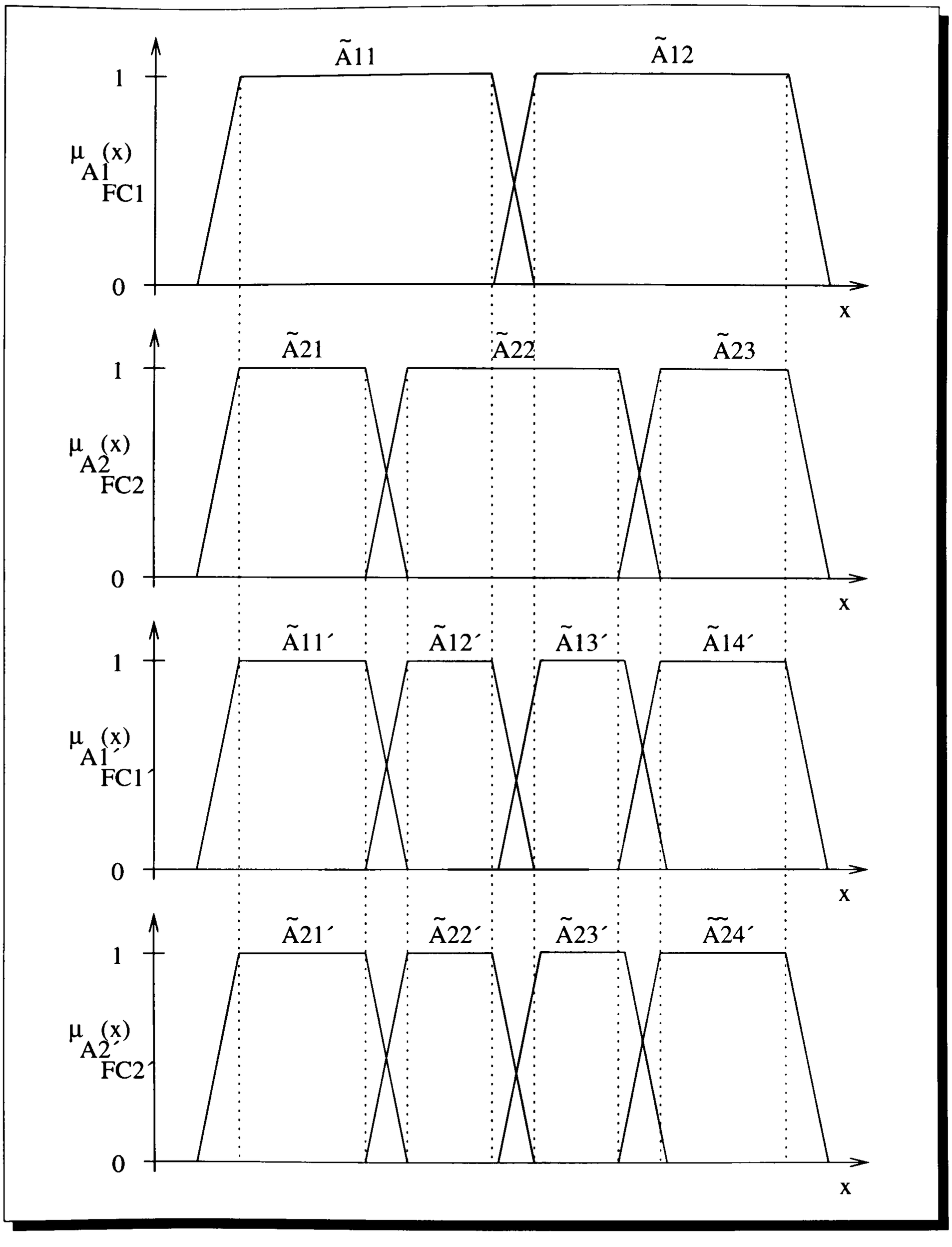


Figure 5.4: Synchronization of the Fuzzy Input Intervals of Two Fuzzy Curves

5.5 Interaction of Fuzzy Curves

Interaction of variables is an important aspect when using mathematical operators. A non interactive division may not contain zero in the denominator whereas a strongly interactive division may. Thus it is necessary to take into account the interaction of Fuzzy Curves. The fuzzy sets are represented by α-cut sets which reduces the fuzzy arithmetic to interval arithmetic.

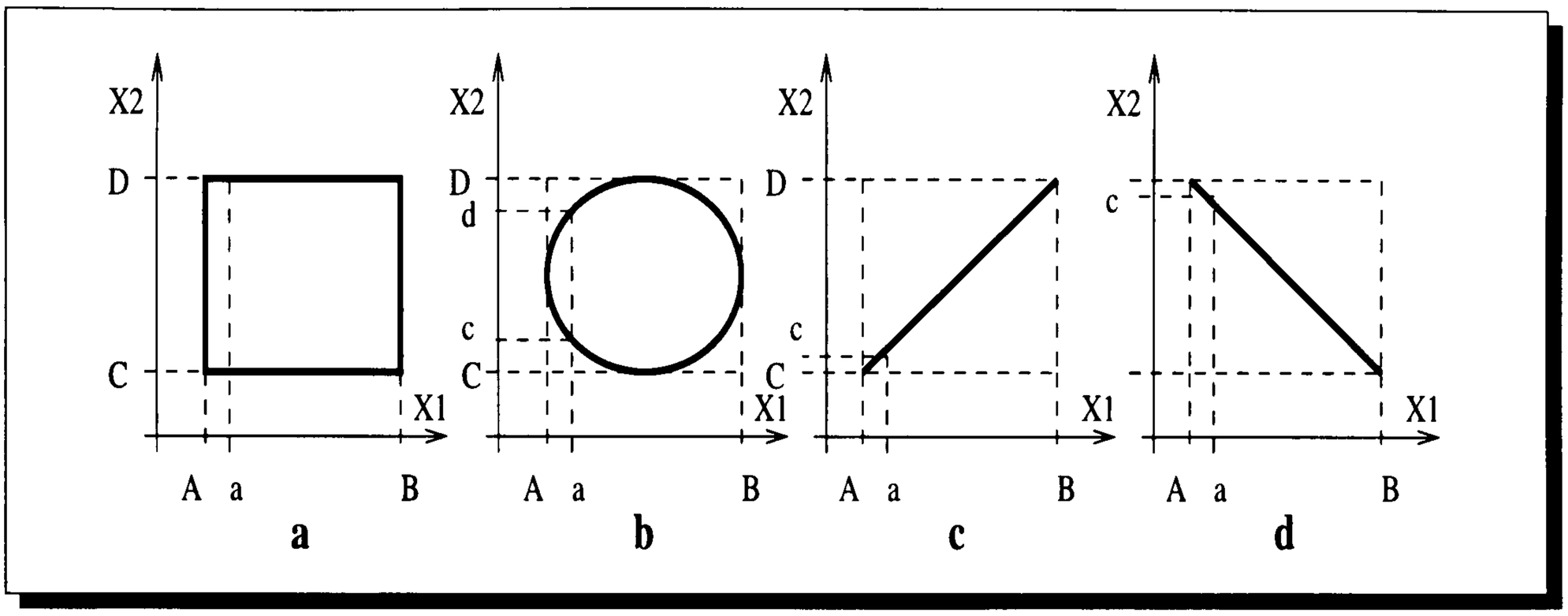


Figure 5.5: Interaction Between Two Interval Variables

Two variables, whose values are, respectively, the two intervals $I_1 = [A, B]$ and $I_2 = [C, D]$, are said to be non interacting if there is no relation between the values they can take into their domains. In other words (Fig.5.5 a), if $x_1 = a \in I_1$, x_2 is not bound to any specific value into I_2 . In the opposite case (Fig.5.5 b,c,d), they are said to be interacting, since $x_1 = a$ implies that x_2 ranges over a specific interval $[c, d]$ contained in I_2. Then, intervals I_1 and I_2 do not represent univocally the approximate relationship existing between the variables x_1 and x_2. J. Buckley and W. Siler [Buckley and Siler, 1988] distinguished interaction into: positively associated, strongly positively associated, negative associated, strongly negative associated variables; the interaction can be defined. In Fig.5.5 c the two variables x_1 and x_2 are strongly positively interacting which maps $x_1 = a \in I_1$ to a single value $x_2 = c \in I_2$. Variables strongly negative interacting are shown in Fig.5.5

d. Any method able to compute the correct range of a function of intervals may be useless if its arguments are interacting. To define the interaction between two variables without additional information is not possible. Functions which contain multiple occurrences of the same variable treat them as different variables, and the resulting interval will be widened. This problem is well-known and several solutions have been discussed to fully solve it under particular conditions discussed in [Dong and Shah, 1987], [Wood et al., 1992], and [Yang et al., 1993].

With fuzzy sets defined through their membership function, the uncertainty involved in values, signals, and models can be considered. The membership functions are defined by intuition of the circuit designer and are by no means correct in any case. It is not necessary to waste computational time in the computation of extreme points, as in the Yang-Yao-Deweg Algorithm [Yang et al., 1993]. On the other hand sometimes the designer knows exactly that an operation must be an interactive one, because of the knowledge of the kind of terms it relates.

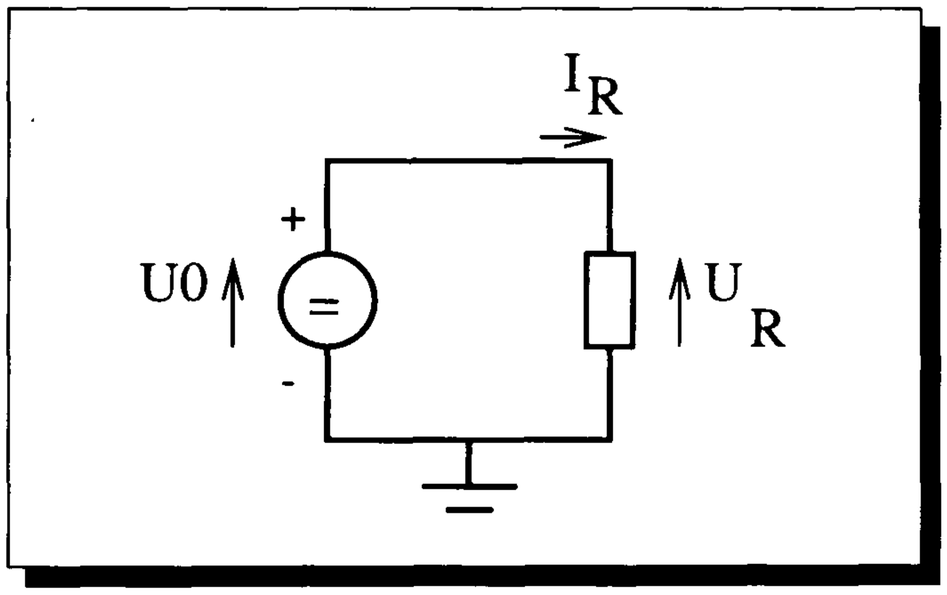


Figure 5.6: Simple Resistor Circuit

Fig. 5.6 is a simple resistor circuit with a voltage source. Suppose the voltage drop at the resistor U_R and the current through the resistor I_R are known fuzzy values. Every circuit designer knows that the resistor is calculated by: $R = \frac{U_R}{I_R}$ and U and I are strongly interactive variables. Not considering the temperature influence to the resistor value, for a particular voltage and current only a particular resistor appears.

In this thesis only the non interaction and the strongly positive/negative interaction between variables is defined. Other interactions are possible to define but out of the scope of this thesis. Besides the strongly positive/negative interactive and the non interactive variables it is difficult to define other inter-activities between variables by designers who have no knowledge about fuzzy logic. Further during the design process there is no need to build a model with weak interacting variables. The designer can build a model with additional constraints that has the same simulation results.

5.6 Operator Notation Overview

In this thesis the notation of the operators are defined as follows:

Operator	Sign of Operator
non interactive operator	$\widetilde{operator}$
strongly positive interactive operator	$\overrightarrow{operator}$
strongly negative interactive operator	$\overleftrightarrow{operator}$

5.7 Addition of Fuzzy Curves

The addition operator is a linear operator. Linear operators map the left/right boundary number of two intervals to a new left/right boundary number of an interval (e.g. $[-3,4]+[-2,1]=[-5,5]$) whether the numbers are negative or not. Nonlinear operators (e.g. multiplication) might map two left boundary numbers to a new right boundary number (e.g. $[-3,4]*[-2,1]=[-8,6]$). Strongly positive interactive and non interactive addition operator of Fuzzy Curves are linear operators too. Both operators combine the left/right boundary functions to new left/right boundary functions of the Fuzzy Curves. Therefore strongly positive interactive and the non interactive addition of two Fuzzy Curves are defined equally.

Definition 5.37 (*Addition of Fuzzy Curves*): Let FC_1 and FC_2 be two Fuzzy Curves and $TFF_1(i)$ and $TFF_2(i)$ be their Temporal Fuzzifying Functions. The addition of the two synchronous Fuzzy Curves is done by adding their Temporal Fuzzifying Functions, so we can then write for *non interactive addition*

$$FC_1 \tilde{+} FC_2 = TFF_1(i) \tilde{+} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.15}$$

$$\tilde{R}^i_{TFF_1(i)\tilde{+}TFF_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \tilde{+} \tilde{f}_{2i}(x) \tag{5.16}$$

for *strongly positive interactive addition*

$$FC_1 \overset{\Rightarrow}{+} FC_2 = TFF_1(i) \overset{\Rightarrow}{+} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.17}$$

$$\tilde{R}^i_{TFF_1(i)\overset{\Rightarrow}{+}TFF_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \overset{\Rightarrow}{+} \tilde{f}_{2i}(x) \tag{5.18}$$

for *strongly negative interactive addition*

$$FC_1 \overset{\rightleftarrows}{+} FC_2 = TFF_1(i) \overset{\rightleftarrows}{+} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.19}$$

$$\tilde{R}^i_{TFF_1(i)\overset{\rightleftarrows}{+}TFF_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \overset{\rightleftarrows}{+} \tilde{f}_{2i}(x) \tag{5.20}$$

Definition 5.38 (*Non Interactive Addition of Temporal Fuzzifying Functions*): The addition of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the addition of their two membership functions, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \; with \; i = 1, 2, ..., n \tag{5.21}$$

then it can be written

$$\mu_{TFF_1 \tilde{+} TFF_2}(x,y) =$$

$$\begin{cases} 0 & if \quad y \leq (f1^-_{>0.0}(x) + f2^-_{>0.0}(x)) \\ L(x,y) & if \quad (f1^-_{>0.0}(x) + f2^-_{>0.0}(x)) < y < (f1^-_{1.0}(x) + f2^-_{1.0}(x)) \\ 1 & if \quad (f1^-_{1.0}(x) + f2^-_{1.0}(x)) \leq y \leq (f1^+_{1.0}(x) + f2^+_{1.0}(x)) \\ R(x,y) & if \quad (f1^+_{1.0}(x) + f2^+_{1.0}(x)) < y < (f1^+_{>0.0}(x) + f2^+_{>0.0}(x)) \\ 0 & if \quad (f1^+_{>0.0}(x) + f2^+_{>0.0}(x)) \leq y \end{cases} \tag{5.22}$$

with:

$$L(x,y) = \frac{y - (f1^-_{>0.0}(x) + f2^-_{>0.0}(x))}{(f1^-_{1.0}(x) + f2^-_{1.0}(x)) - (f1^-_{>0.0}(x) + f2^-_{>0.0}(x))} \tag{5.23}$$

$$R(x,y) = \frac{(f1^+_{>0.0}(x) + f2^+_{>0.0}(x)) - y}{(f1^+_{>0.0}(x) + f2^+_{>0.0}(x)) - (f1^+_{1.0}(x) + f2^+_{1.0}(x))} \tag{5.24}$$

The non interactive and the strongly positive interactive addition of temporal fuzzifying functions is equal.

Definition 5.39 (*Strongly Positive Interactive Addition of Temporal Fuzzifying Functions*): Equal to non interactive addition.

$$\mu_{TFF_1 \tilde{\tilde{+}} TFF_2}(x,y) = \mu_{TFF_1 \tilde{+} TFF_2}(x,y) \tag{5.25}$$

Definition 5.40 (*Strongly Negative Interactive Addition of Temporal Fuzzifying Functions*): The addition of two Temporal Fuzzifying Functions TFF_1 and TFF_2 strongly negative interacting is the addition of their two membership functions, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \; with \; i = 1, 2, ..., n \tag{5.26}$$

then it can be written

$$\mu_{TFF_1 \stackrel{=}{+} TFF_2}(x,y) =$$

$$\begin{cases} 0 & if \quad y \leq (f1^{-}_{>0.0}(x) + f2^{+}_{>0.0}(x)) \\ L(x,y) & if \quad (f1^{-}_{>0.0}(x) + f2^{+}_{>0.0}(x)) < y < (f1^{-}_{1.0}(x) + f2^{+}_{1.0}(x)) \\ 1 & if \quad (f1^{-}_{1.0}(x) + f2^{+}_{1.0}(x)) \leq y \leq (f1^{+}_{1.0}(x) + f2^{-}_{1.0}(x)) \\ R(x,y) & if \quad (f1^{+}_{1.0}(x) + f2^{-}_{1.0}(x)) < y < (f1^{+}_{>0.0}(x) + f2^{-}_{>0.0}(x)) \\ 0 & if \quad (f1^{+}_{>0.0}(x) + f2^{-}_{>0.0}(x)) \leq y \end{cases} \tag{5.27}$$

with:

$$L(x,y) = \frac{y - (f1^{-}_{>0.0}(x) + f2^{+}_{>0.0}(x))}{(f1^{-}_{1.0}(x) + f2^{+}_{1.0}(x)) - (f1^{-}_{>0.0}(x) + f2^{+}_{>0.0}(x))} \tag{5.28}$$

$$R(x,y) = \frac{(f1^{+}_{>0.0}(x) + f2^{-}_{>0.0}(x)) - y}{(f1^{+}_{>0.0}(x) + f2^{-}_{>0.0}(x)) - (f1^{+}_{1.0}(x) + f2^{-}_{1.0}(x))} \tag{5.29}$$

5.7.1 Example: Addition of Fuzzy Curves

Fig. 5.9 shows the addition of the two fuzzy curves FC1 and FC2. The Fuzzy Curves are defined by three α-cut levels:

α-cut level	data file
> 0.0	fc00.dat
0.6	fc06.dat
1.0	fc10.dat

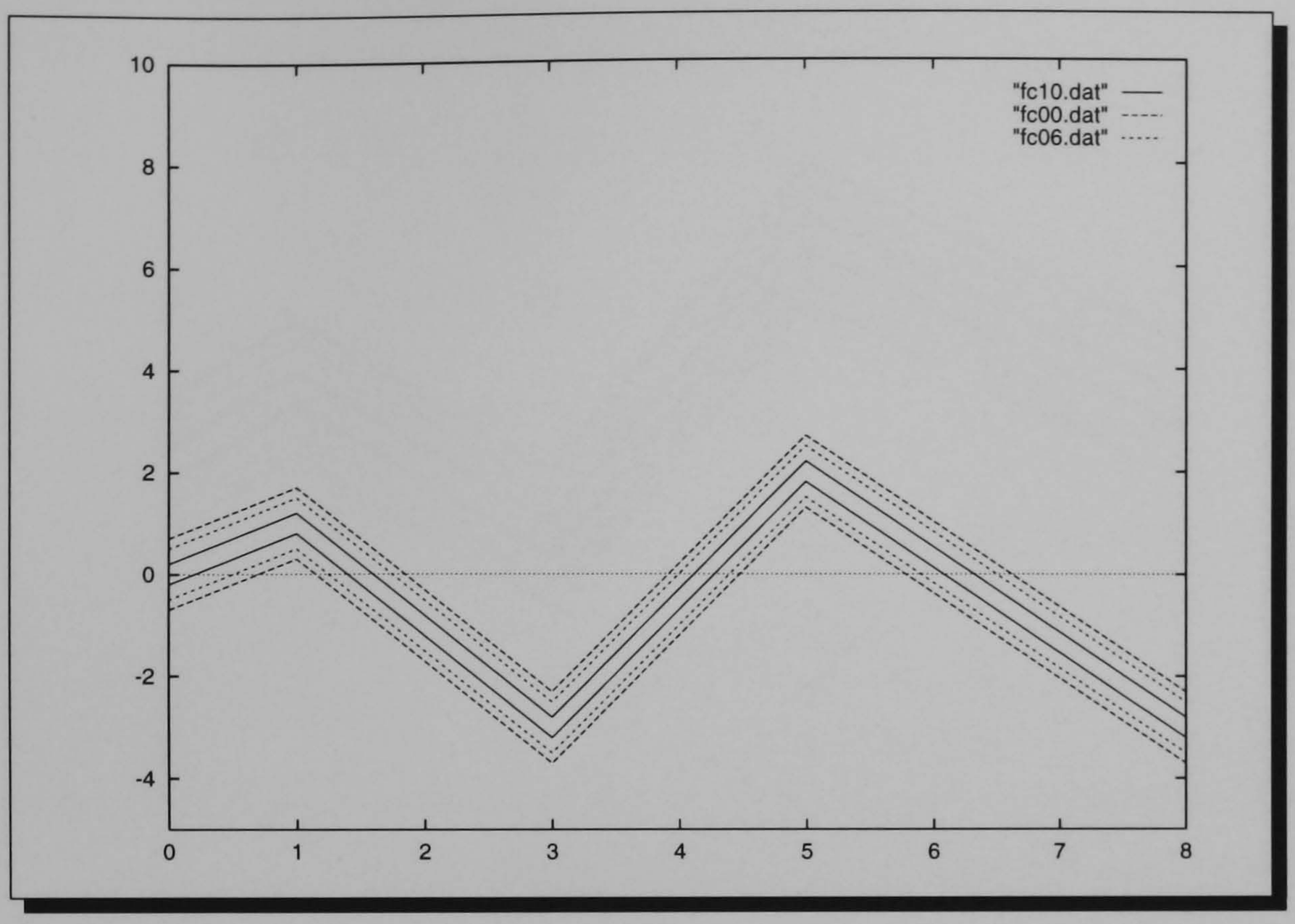


Figure 5.7: Non Interactive Addition: Fuzzy Curve FC_1

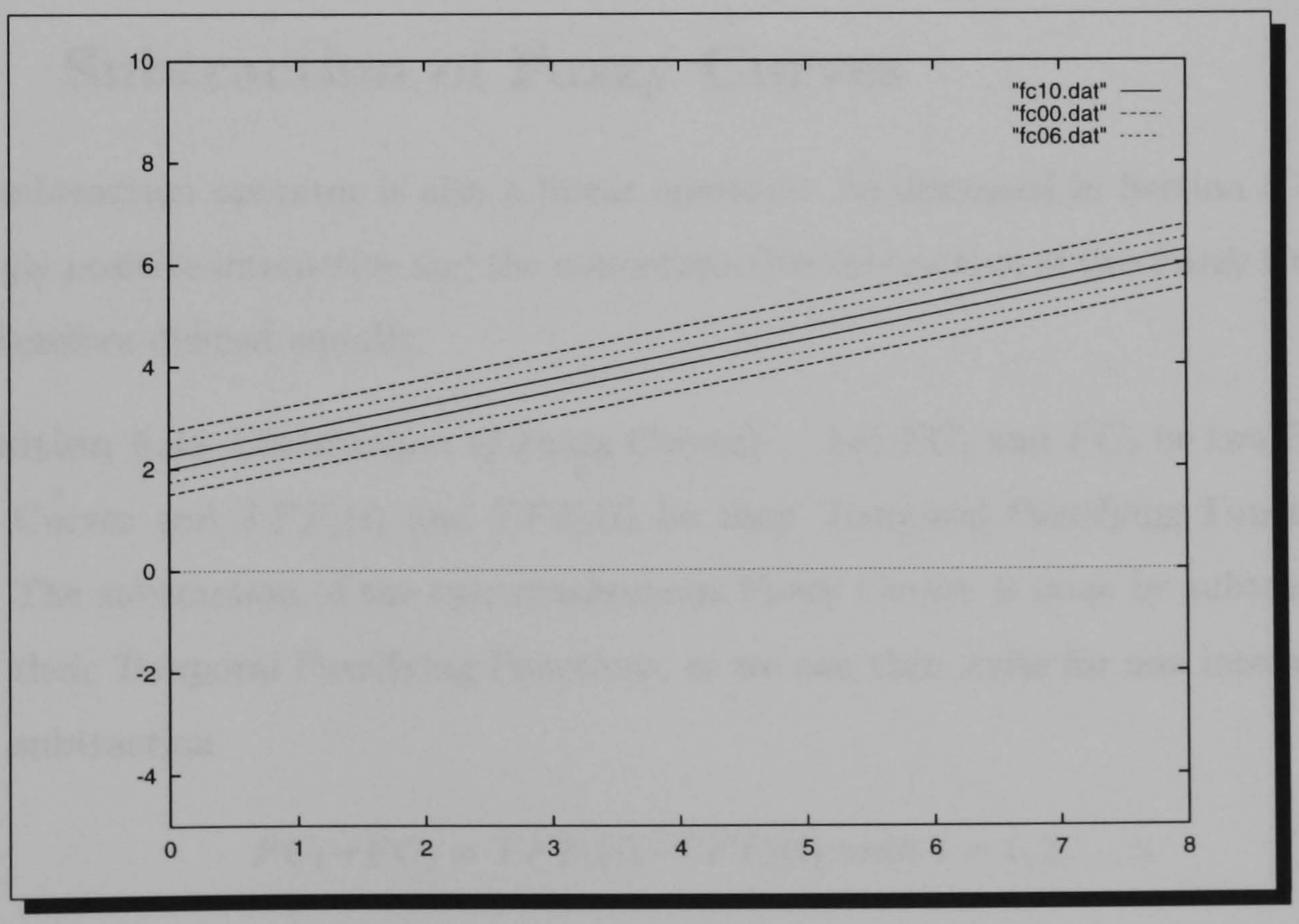


Figure 5.8: Non Interactive Addition: Fuzzy Curve FC_2

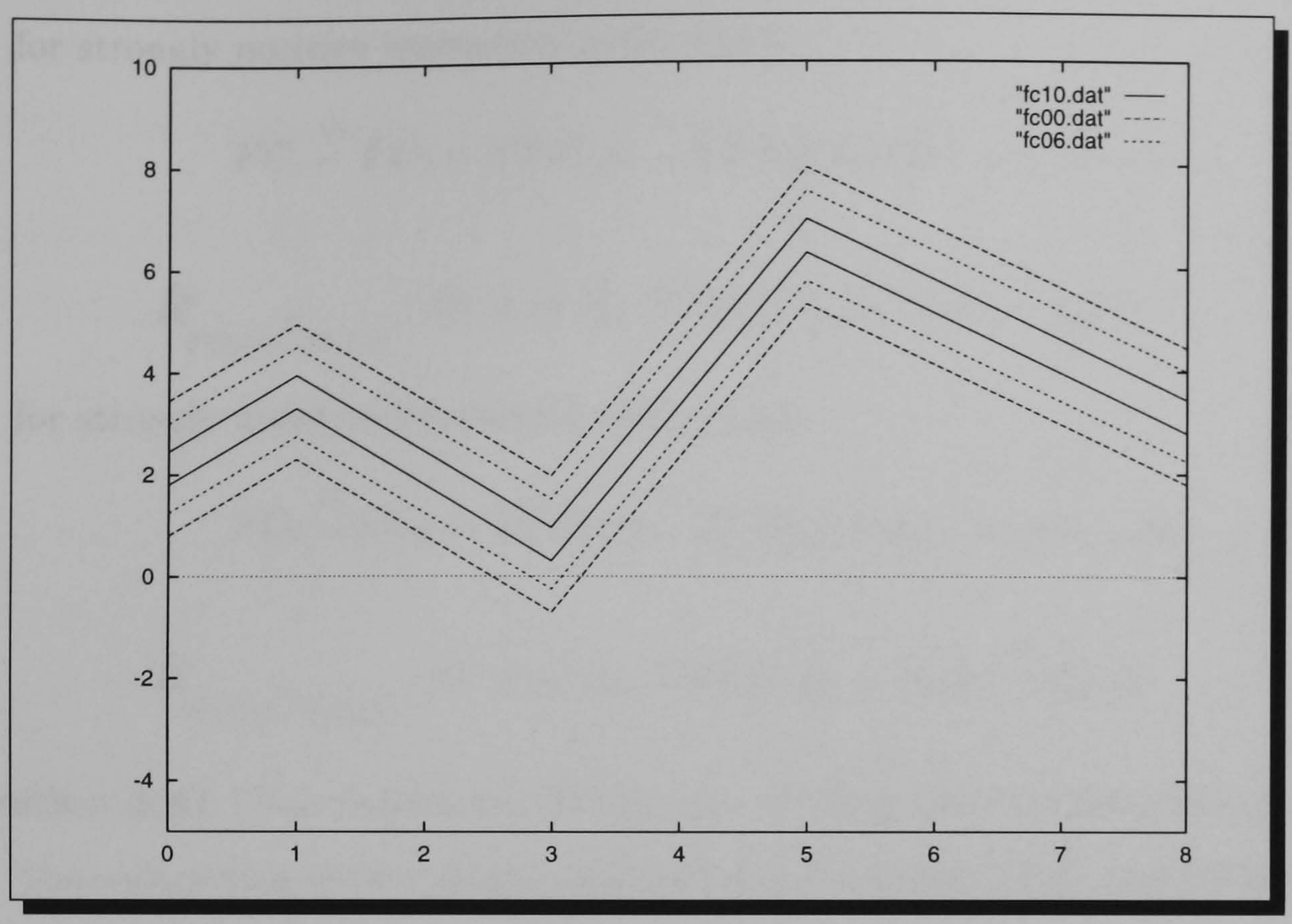


Figure 5.9: Non Interactive Addition of FC_1 and FC_2

5.8 Subtraction of Fuzzy Curves

The subtraction operator is also a linear operator. As discussed in Section 5.7 the strongly positive interactive and the non interactive subtraction of two Fuzzy Curves are therefore defined equally.

Definition 5.41 (*Subtraction of Fuzzy Curves*): Let FC_1 and FC_2 be two Fuzzy Curves and $TFF_1(i)$ and $TFF_2(i)$ be their Temporal Fuzzifying Functions. The subtraction of the two synchronous Fuzzy Curves is done by subtracting their Temporal Fuzzifying Functions, so we can then write for non interactive subtraction

$$FC_1 \tilde{-} FC_2 = TFF_1(i) \tilde{-} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.30}$$

$$\tilde{R}^i_{FC_1(i) \tilde{-} FC_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \tilde{-} \tilde{f}_{2i}(x) \tag{5.31}$$

for strongly positive interactive subtraction

$$FC_1 \overset{\rightrightarrows}{-} FC_2 = TFF_1(i) \overset{\rightrightarrows}{-} TFF_2(i) \ with \ i = 1, 2, ..., n \tag{5.32}$$

$$\tilde{R}^i_{FC_1(i) \overset{\rightrightarrows}{-} FC_2(i)} : IF \ x \ is \ \tilde{A}_i, \ THEN \ \tilde{y}_i \ is \ \tilde{f}_{1i}(x) \overset{\rightrightarrows}{-} \tilde{f}_{2i}(x) \tag{5.33}$$

for strongly negative interactive subtraction

$$FC_1 \overset{\rightleftarrows}{-} FC_2 = TFF_1(i) \overset{\rightleftarrows}{-} TFF_2(i) \ with \ i = 1, 2, ..., n \tag{5.34}$$

$$\tilde{R}^i_{FC_1(i) \overset{\rightleftarrows}{-} FC_2(i)} : IF \ x \ is \ \tilde{A}_i, \ THEN \ \tilde{y}_i \ is \ \tilde{f}_{1i}(x) \overset{\rightleftarrows}{-} \tilde{f}_{2i}(x) \tag{5.35}$$

Definition 5.42 (*Non Interactive Subtraction of Temporal Fuzzifying Functions*): The subtraction of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the subtraction of their two membership functions, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \ with \ i = 1, 2, ..., n \tag{5.36}$$

then it can be written

$$\mu_{TFF_1 \tilde{-} TFF_2}(x, y) =$$

$$\begin{cases} 0 & if \quad y \leq (f1^-_{>0.0}(x) - f2^-_{>0.0}(x)) \\ L(x,y) & if \quad (f1^-_{>0.0}(x) - f2^-_{>0.0}(x)) < y < (f1^-_{1.0}(x) - f2^-_{1.0}(x)) \\ 1 & if \quad (f1^-_{1.0}(x) - f2^-_{1.0}(x)) \leq y \leq (f1^+_{1.0}(x) - f2^+_{1.0}(x)) \\ R(x,y) & if \quad (f1^+_{1.0}(x) - f2^+_{1.0}(x)) < y < (f1^+_{>0.0}(x) - f2^+_{>0.0}(x)) \\ 0 & if \quad (f1^+_{>0.0}(x) - f2^+_{>0.0}(x)) \leq y \end{cases} \tag{5.37}$$

with:

$$L(x, y) = \frac{y - (f1^-_{>0.0}(x) - f2^-_{>0.0}(x))}{(f1^-_{1.0}(x) - f2^-_{1.0}(x)) - (f1^-_{>0.0}(x) - f2^-_{>0.0}(x))} \tag{5.38}$$

$$R(x, y) = \frac{(f1^+_{>0.0}(x) - f2^+_{>0.0}(x)) - y}{(f1^+_{>0.0}(x) - f2^+_{>0.0}(x)) - (f1^+_{1.0}(x) - f2^+_{1.0}(x))} \tag{5.39}$$

The non interactive and the strongly positive interactive subtraction of Temporal Fuzzifying Functions is equal.

Definition 5.43 (*Strongly Positive Interactive Subtraction of Temporal Fuzzifying Functions*): Equal to non interactive subtraction.

$$\mu_{TFF_1 \dot{-} TFF_2}(x,y) = \mu_{TFF_1 \tilde{-} TFF_2}(x,y) \tag{5.40}$$

Definition 5.44 (*Strongly Negative Interactive Subtraction of Temporal Fuzzifying Functions*): The subtraction of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the subtraction of their two membership functions, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \; with \; i = 1, 2, ..., n \tag{5.41}$$

then it can be written

$$\mu_{TFF_1 \tilde{-} TFF_2}(x,y) =$$

$$\begin{cases} 0 & if \quad y \leq (f1^{-}_{>0.0}(x) - f2^{+}_{>0.0}(x)) \\ L(x,y) & if \quad (f1^{-}_{>0.0}(x) - f2^{+}_{>0.0}(x)) < y < (f1^{-}_{1.0}(x) - f2^{+}_{1.0}(x)) \\ 1 & if \quad (f1^{-}_{1.0}(x) - f2^{+}_{1.0}(x)) \leq y \leq (f1^{+}_{1.0}(x) - f2^{-}_{1.0}(x)) \\ R(x,y) & if \quad (f1^{+}_{1.0}(x) - f2^{-}_{1.0}(x)) < y < (f1^{+}_{>0.0}(x) - f2^{-}_{>0.0}(x)) \\ 0 & if \quad (f1^{+}_{>0.0}(x) - f2^{-}_{>0.0}(x)) \leq y \end{cases} \tag{5.42}$$

with:

$$L(x,y) = \frac{y - (f1^{-}_{>0.0}(x) - f2^{+}_{>0.0}(x))}{(f1^{-}_{1.0}(x) - f2^{+}_{1.0}(x)) - (f1^{-}_{>0.0}(x) - f2^{+}_{>0.0}(x))} \tag{5.43}$$

$$R(x,y) = \frac{(f1^{+}_{>0.0}(x) - f2^{-}_{>0.0}(x)) - y}{(f1^{+}_{>0.0}(x) - f2^{-}_{>0.0}(x)) - (f1^{+}_{1.0}(x) - f2^{-}_{1.0}(x))} \tag{5.44}$$

5.8.1 Example: Subtraction of Fuzzy Curves

Fig. 5.12 shows the subtraction of the two Fuzzy Curves FC1 and FC2. The Fuzzy Curves are defined by three α-cut levels:

α-cut level	data file
> 0.0	fc00.dat
0.6	fc06.dat
1.0	fc10.dat

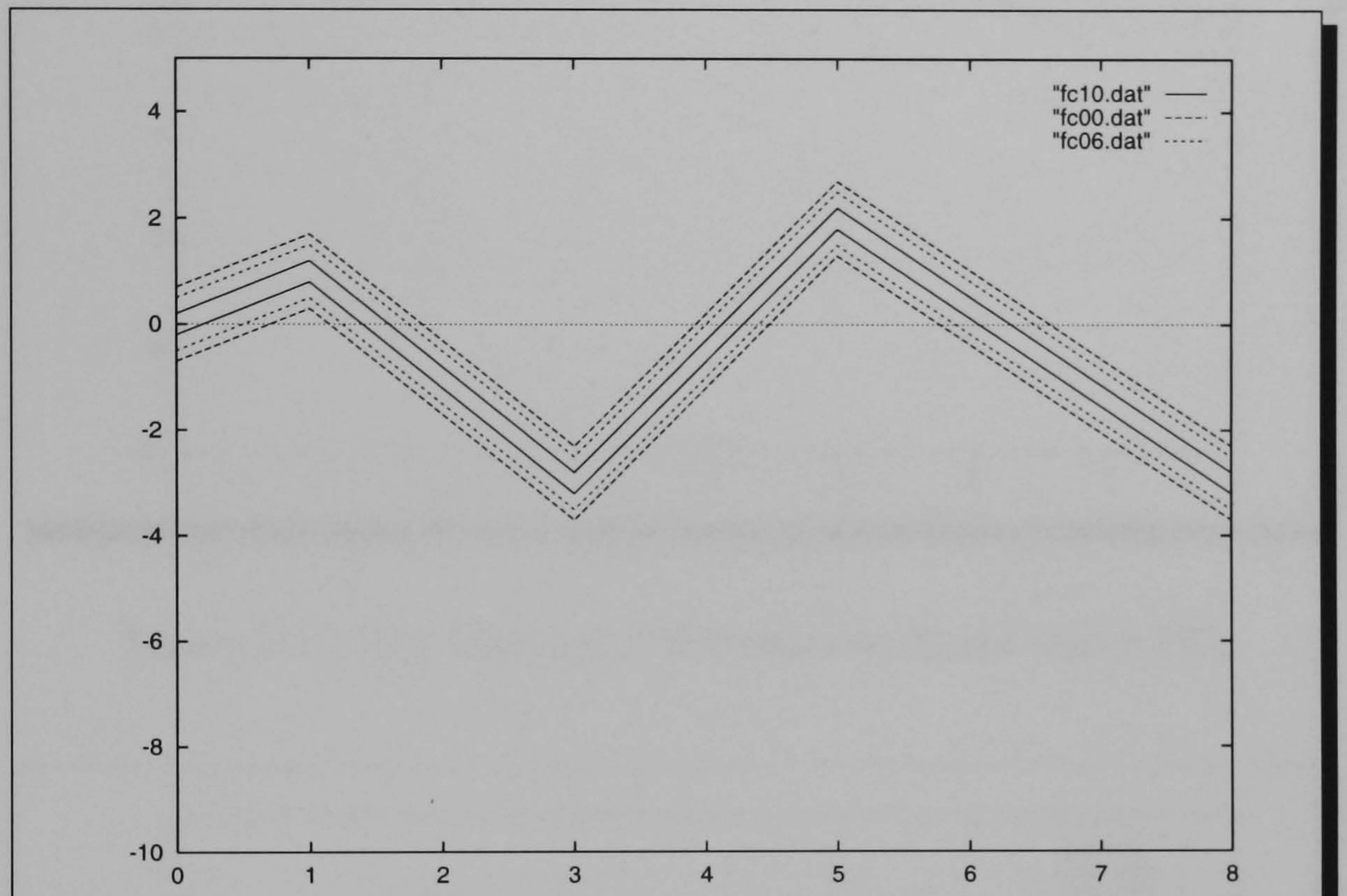


Figure 5.10: Non Interactive Subtraction: Fuzzy Curve FC_1

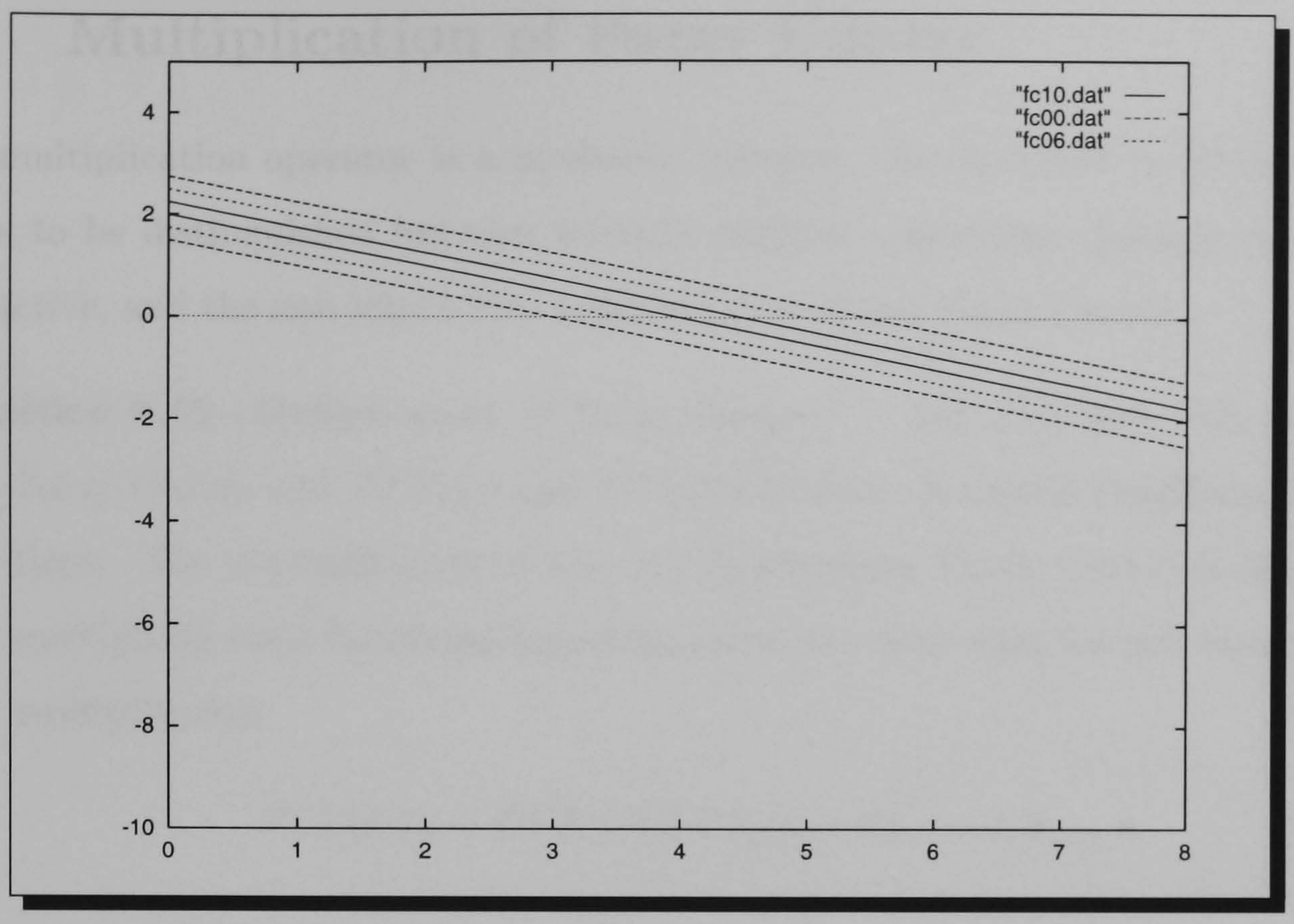


Figure 5.11: Non Interactive Subtraction: Fuzzy Curve FC_2

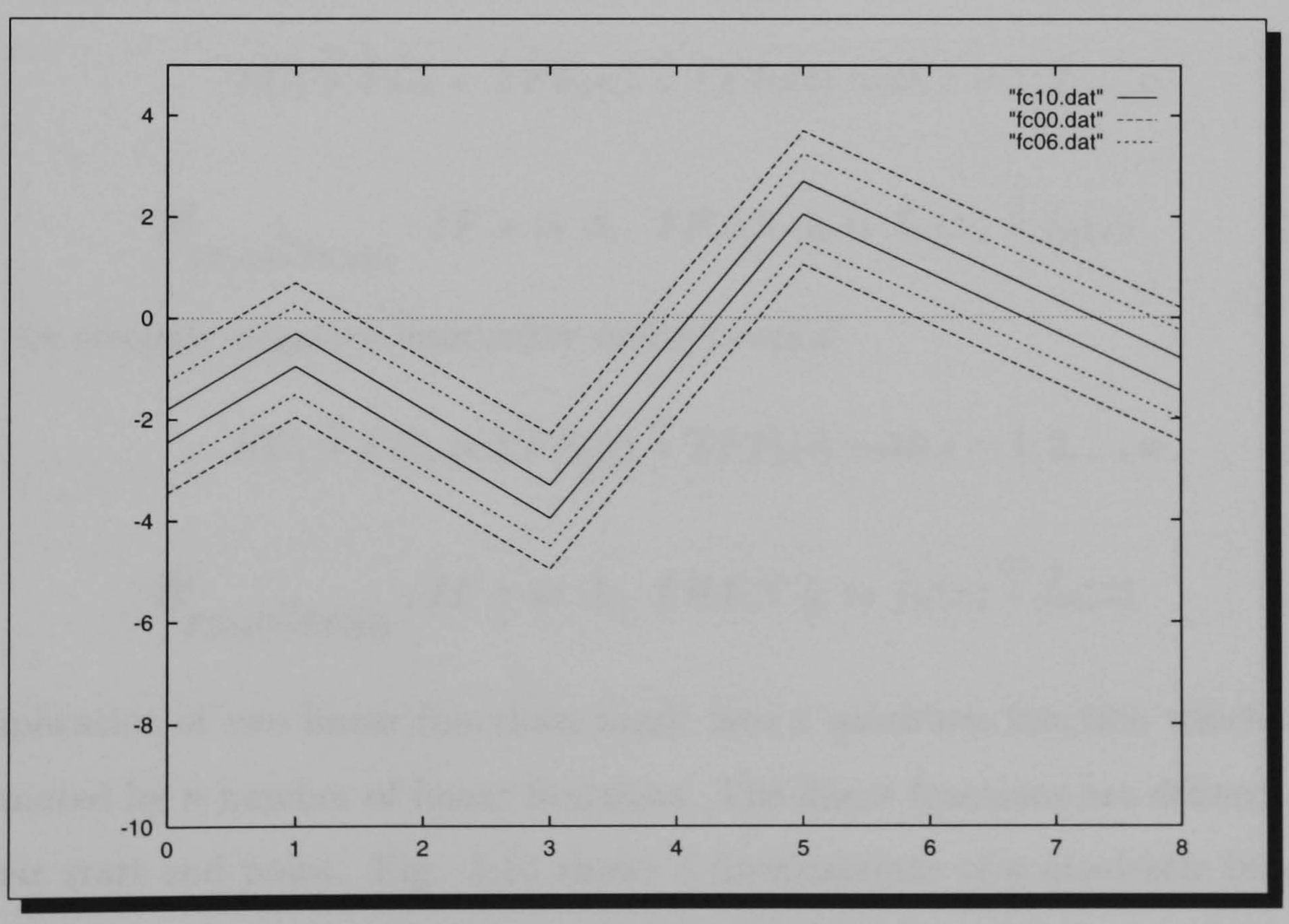


Figure 5.12: Non Interactive Subtraction of FC_1 and FC_2

5.9 Multiplication of Fuzzy Curves

The multiplication operator is a nonlinear operator. As discussed in Section 5.7, it has to be distinguished between strongly positive interactive, strongly negative interactive, and the non interactive multiplication of two Fuzzy Curves.

Definition 5.45 (*Multiplication of Fuzzy Curves*): Let FC_1 and FC_2 be two Fuzzy Curves and $TFF_1(i)$ and $TFF_2(i)$ be their Temporal Fuzzifying Functions. The multiplication of the two synchronous Fuzzy Curves is done by multiplying their fuzzifying functions, so we can then write for non interactive multiplication

$$FC_1 \tilde{*} FC_2 = TFF_1(i) \tilde{*} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.45}$$

$$\tilde{R}^i_{FC_1(i) \tilde{*} FC_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \tilde{*} \tilde{f}_{2i}(x) \tag{5.46}$$

for strongly positive interactive multiplication

$$FC_1 \overset{\rightrightarrows}{*} FC_2 = TFF_1(i) \overset{\rightrightarrows}{*} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.47}$$

$$\tilde{R}^i_{FC_1(i) \overset{\rightrightarrows}{*} FC_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \overset{\rightrightarrows}{*} \tilde{f}_{2i}(x) \tag{5.48}$$

for strongly negative interactive multiplication

$$FC_1 \overset{\rightleftarrows}{*} FC_2 = TFF_1(i) \overset{\rightleftarrows}{*} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.49}$$

$$\tilde{R}^i_{FC_1(i) \overset{\rightleftarrows}{*} FC_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \overset{\rightleftarrows}{*} \tilde{f}_{2i}(x) \tag{5.50}$$

Multiplication of two linear functions result into a quadratic function which is approximated by n number of linear functions. The linear functions are defined exact at their start end point. Fig. 5.13 shows 3 linearisations of a quadratic function; $n = 2$ (2 point approximation), $n = 3$ (3 point approximation), $n = 5$ (5 point approximation).

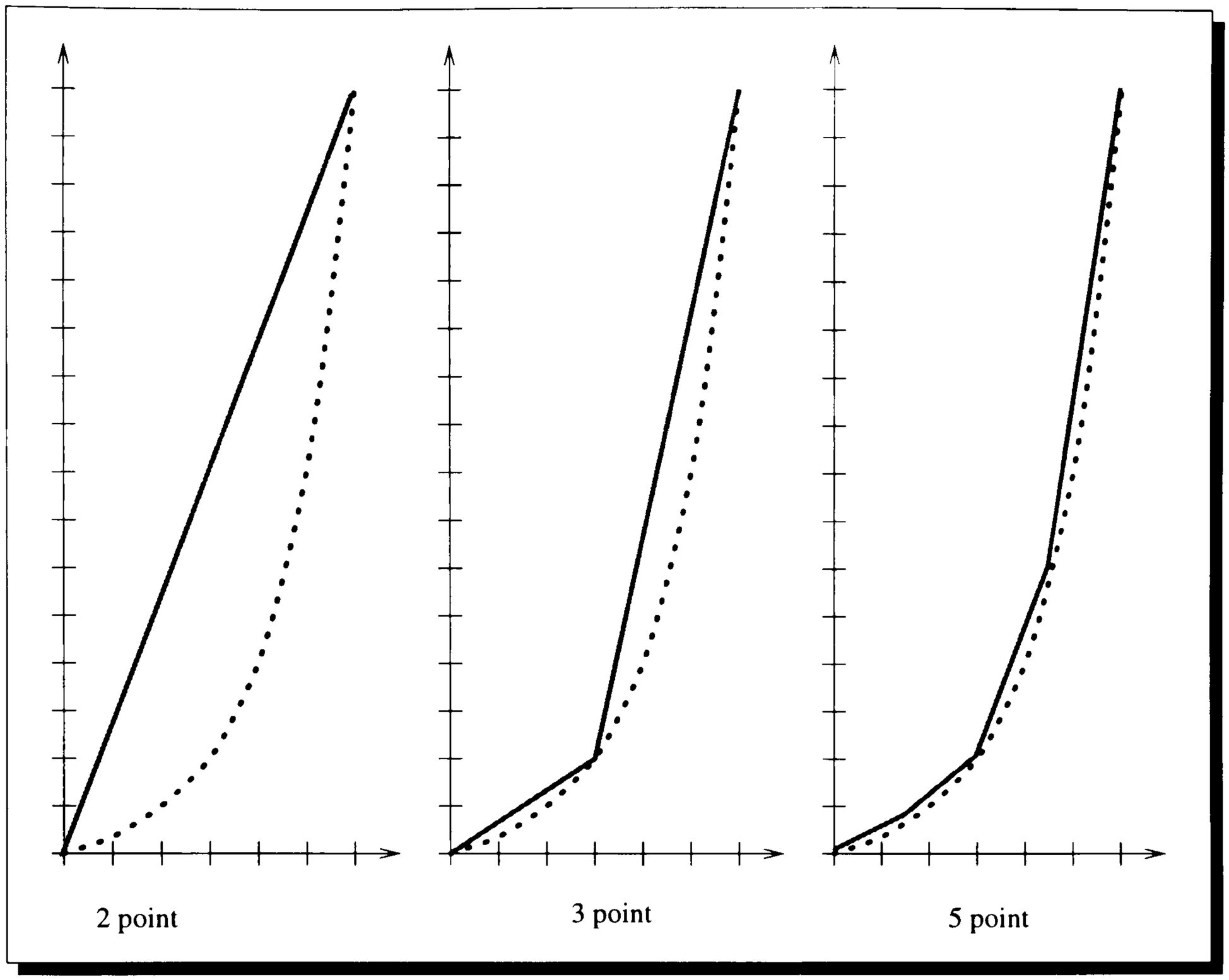


Figure 5.13: Linearization: 2 Point, 3 Point, 5 Point Approximation

The non interactive and the strongly positive/negative interactive multiplication of Temporal Fuzzifying Functions have to be defined differently.

Definition 5.46 (*Non Interactive Multiplication of Temporal Fuzzifying Functions*): The non interactive multiplication of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the multiplication of their two membership functions non interactively, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A}_1{}'(i) = \tilde{A}_2{}'(i) \; with \; i = 1, 2, ..., n \tag{5.51}$$

then it can be written

$$\mu_{TFF_1 \tilde{*} TFF_2}(x,y) = \begin{cases} 0 & if \quad y \leq min(TFF - Set_{>0.0}) \\ L(x,y) & if \quad min(TFF - Set_{>0.0}) < y < min(TFF - Set_{1.0}) \\ 1 & if \quad min(TFF - Set_{1.0}) \leq y \leq max(TFF - Set_{1.0}) \\ R(x,y) & if \quad max(TFF - Set_{1.0}) < y < max(TFF - Set_{>0.0}) \\ 0 & if \quad max(TFF - Set_{>0.0}) \leq y \end{cases} \tag{5.52}$$

with:

$$\begin{aligned} TFF - Set_{>0.0} = \{ & f1^-_{>0.0}(x) * f2^-_{>0.0}(x), \\ & f1^-_{>0.0}(x) * f2^+_{>0.0}(x), \\ & f1^+_{>0.0}(x) * f2^-_{>0.0}(x), \\ & f1^+_{>0.0}(x) * f2^+_{>0.0}(x)\} \end{aligned} \tag{5.53}$$

$$\begin{aligned} TFF - Set_{1.0} = \{ & f1^-_{1.0}(x) * f2^-_{1.0}(x), \\ & f1^-_{1.0}(x) * f2^+_{1.0}(x), \\ & f1^+_{1.0}(x) * f2^-_{1.0}(x), \\ & f1^+_{1.0}(x) * f2^+_{1.0}(x)\} \end{aligned} \tag{5.54}$$

$$L(x,y) = \frac{y - min(TFF - Set_{>0.0})}{min(TFF - Set_{1.0}) - min(TFF - Set_{>0.0})} \tag{5.55}$$

$$R(x,y) = \frac{max(TFF - Set_{>0.0}) - y}{max(TFF - Set_{>0.0}) - max(TFF - Set_{1.0})} \tag{5.56}$$

5.9.1 Example: Non Interactive Multiplication of Fuzzy Curves

Fig. 5.16 shows a non interactive multiplication of the two Fuzzy Curves FC1 and FC2. There has been used a 3-point approximation. The Fuzzy Curves are defined by three α-cut levels:

α-cut level	data file
> 0.0	fc00.dat
0.6	fc06.dat
1.0	fc10.dat

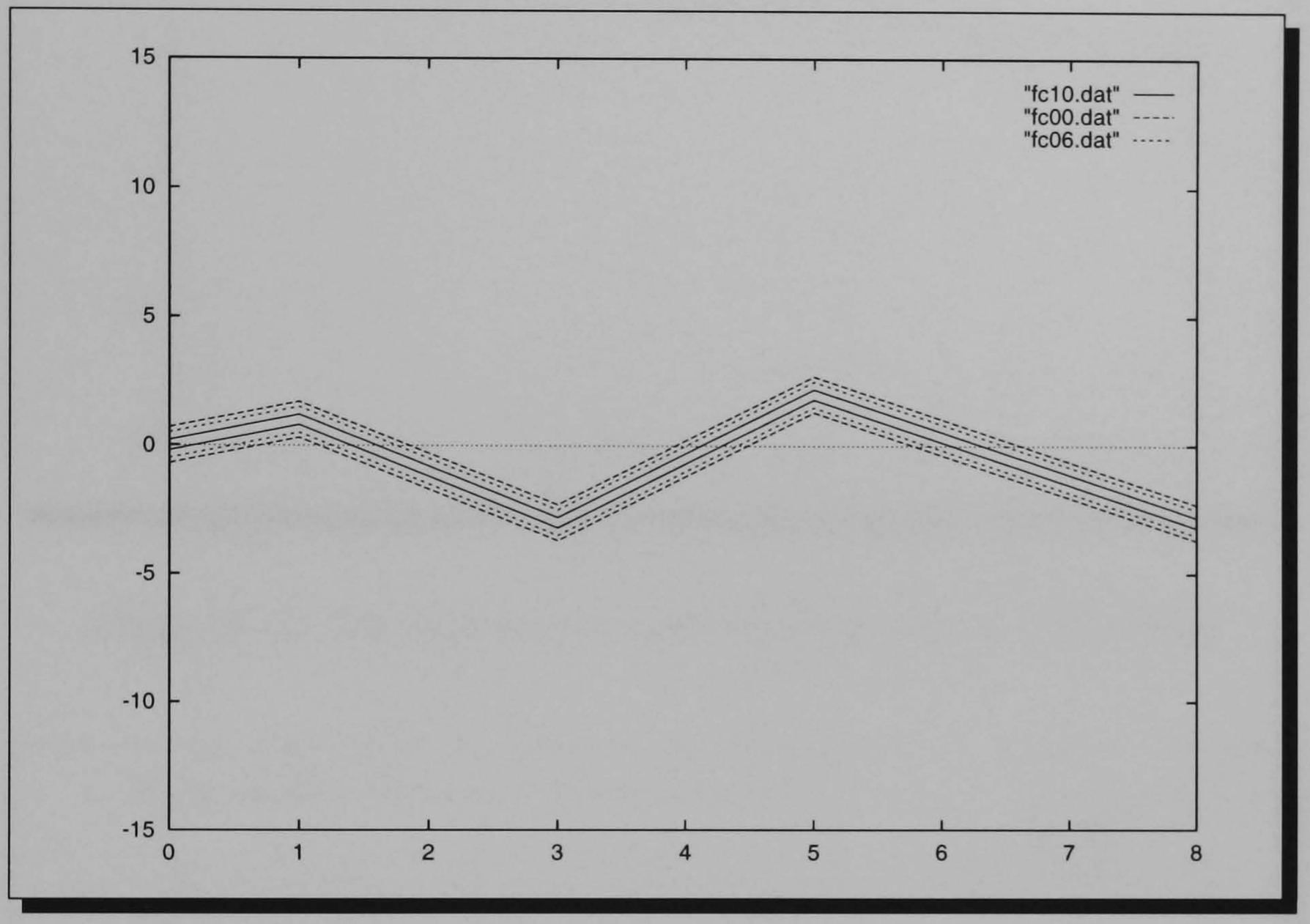


Figure 5.14: Non Interactive Multiplication: Fuzzy Curve FC_1

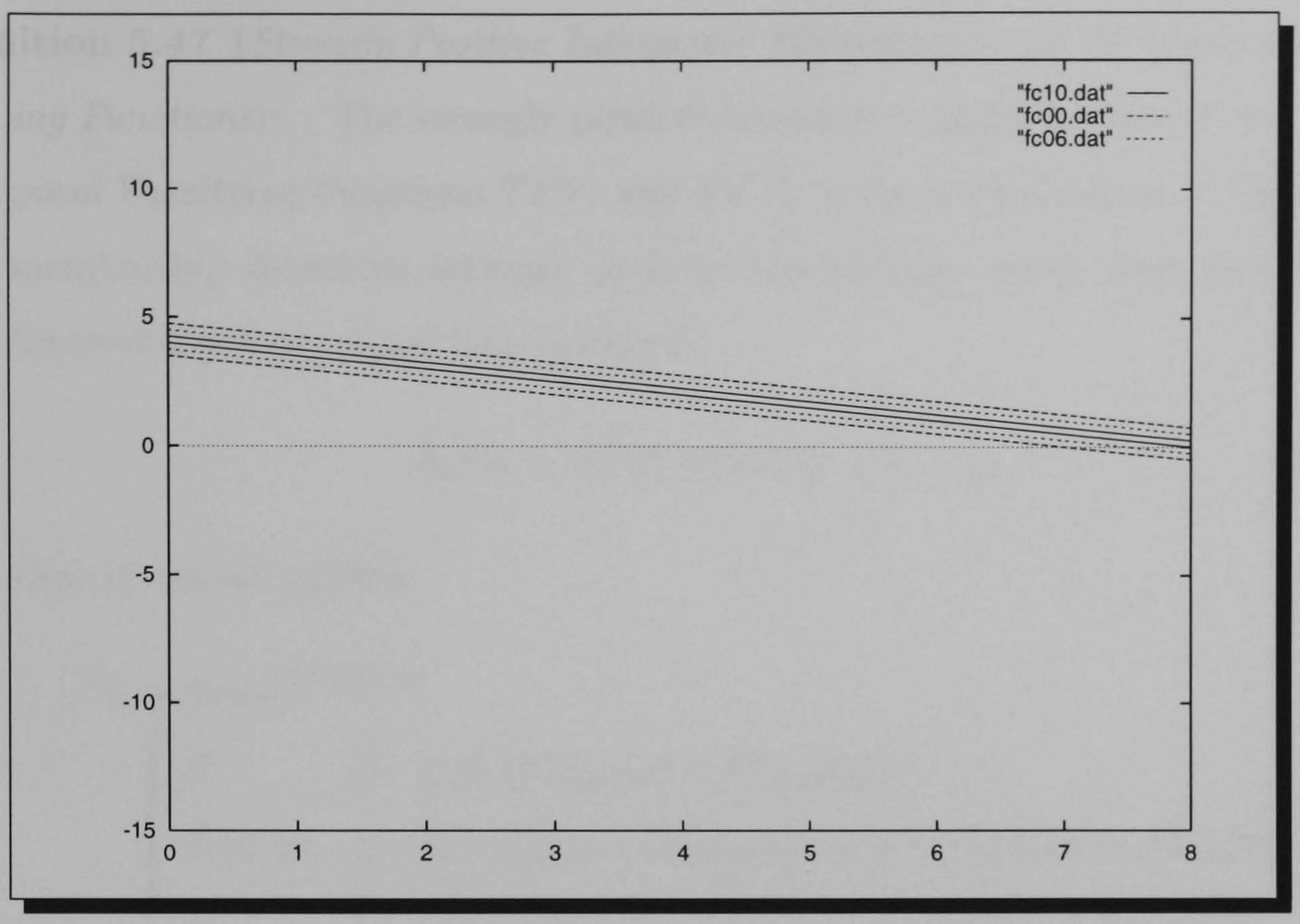


Figure 5.15: Non Interactive Multiplication: Fuzzy Curve FC_2

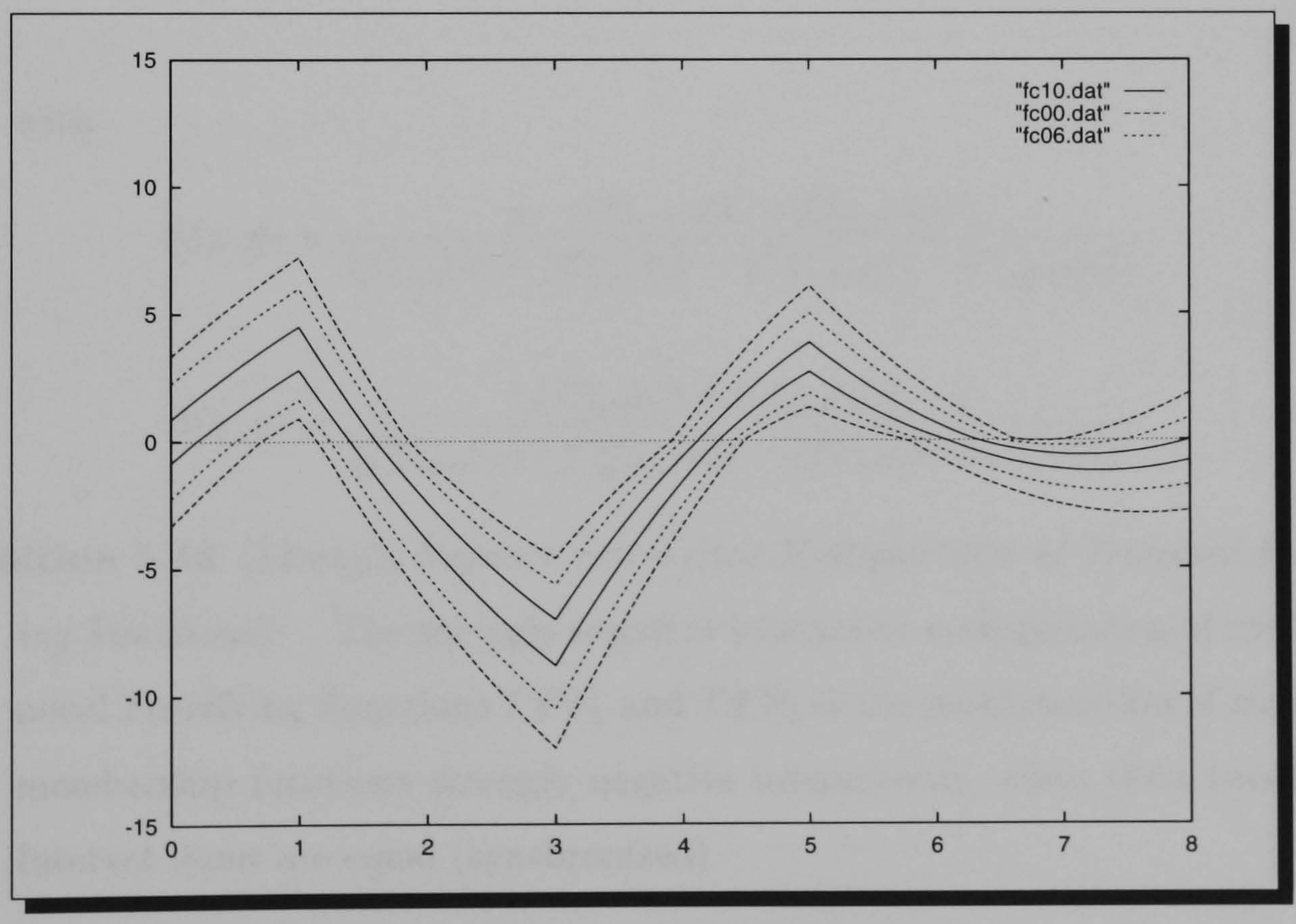


Figure 5.16: Non Interactive Multiplication of FC_1 and FC_2

Definition 5.47 (*Strongly Positive Interactive Multiplication of Temporal Fuzzifying Functions*): The strongly positive interactive multiplication of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the multiplication of their two membership functions strongly positive interactively, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \; with \; i = 1, 2, ..., n \tag{5.57}$$

then it can be written

$$\mu_{TFF_1 \overrightarrow{*} TFF_2}(x, y) =$$

$$\begin{cases} 0 & if \quad y \leq (f1^{-}_{>0.0}(x) * f2^{-}_{>0.0}(x)) \\ L(x,y) & if \quad (f1^{-}_{>0.0}(x) * f2^{-}_{>0.0}(x)) < y < (f1^{-}_{1.0}(x) * f2^{-}_{1.0}(x)) \\ 1 & if \quad (f1^{-}_{1.0}(x) * f2^{-}_{1.0}(x)) \leq y \leq (f1^{+}_{1.0}(x) * f2^{+}_{1.0}(x)) \\ R(x,y) & if \quad (f1^{+}_{1.0}(x) * f2^{+}_{1.0}(x)) < y < (f1^{+}_{>0.0}(x) * f2^{+}_{>0.0}(x)) \\ 0 & if \quad (f1^{+}_{>0.0}(x) * f2^{+}_{>0.0}(x)) \leq y \end{cases} \tag{5.58}$$

with:

$$L(x,y) = \frac{y - (f1^{-}_{>0.0}(x) * f2^{-}_{>0.0}(x))}{(f1^{-}_{1.0}(x) * f2^{-}_{1.0}(x)) - (f1^{-}_{>0.0}(x) * f2^{-}_{>0.0}(x))} \tag{5.59}$$

$$R(x,y) = \frac{(f1^{+}_{>0.0}(x) * f2^{+}_{>0.0}(x)) - y}{(f1^{+}_{>0.0}(x) * f2^{+}_{>0.0}(x)) - (f1^{+}_{1.0}(x) * f2^{+}_{1.0}(x))} \tag{5.60}$$

Definition 5.48 (*Strongly Negative Interactive Multiplication of Temporal Fuzzifying Functions*): The strongly negative interactive multiplication of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the multiplication of their two membership functions strongly negative interactively, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \; with \; i = 1, 2, ..., n \tag{5.61}$$

then it can be written

$$\mu_{TFF_1 \tilde{*} TFF_2}(x,y) =$$

$$\begin{cases} 0 & if \quad y \leq (f1^{-}_{>0.0}(x) * f2^{+}_{>0.0}(x)) \\ L(x,y) & if \quad (f1^{-}_{>0.0}(x) * f2^{+}_{>0.0}(x)) < y < (f1^{-}_{1.0}(x) * f2^{+}_{1.0}(x)) \\ 1 & if \quad (f1^{-}_{1.0}(x) * f2^{+}_{1.0}(x)) \leq y \leq (f1^{+}_{1.0}(x) * f2^{-}_{1.0}(x)) \\ R(x,y) & if \quad (f1^{+}_{1.0}(x) * f2^{-}_{1.0}(x)) < y < (f1^{+}_{>0.0}(x) * f2^{-}_{>0.0}(x)) \\ 0 & if \quad (f1^{+}_{>0.0}(x) * f2^{-}_{>0.0}(x)) \leq y \end{cases} \tag{5.62}$$

with:

$$L(x,y) = \frac{y - (f1^{-}_{>0.0}(x) * f2^{+}_{>0.0}(x))}{(f1^{-}_{1.0}(x) * f2^{+}_{1.0}(x)) - (f1^{-}_{>0.0}(x) * f2^{+}_{>0.0}(x))} \tag{5.63}$$

$$R(x,y) = \frac{(f1^{+}_{>0.0}(x) * f2^{-}_{>0.0}(x)) - y}{(f1^{+}_{>0.0}(x) * f2^{-}_{>0.0}(x)) - (f1^{+}_{1.0}(x) * f2^{-}_{1.0}(x))} \tag{5.64}$$

5.9.2 Example: Strongly Positive Interactive Multiplication of Fuzzy Curves

Fig. 5.19 shows a strongly interactive multiplication of the two Fuzzy Curves FC1 and FC2. There has been used a 3-point approximation. The Fuzzy Curves are defined by three α-cut levels:

α-cut level	data file
> 0.0	fc00.dat
0.6	fc06.dat
1.0	fc10.dat

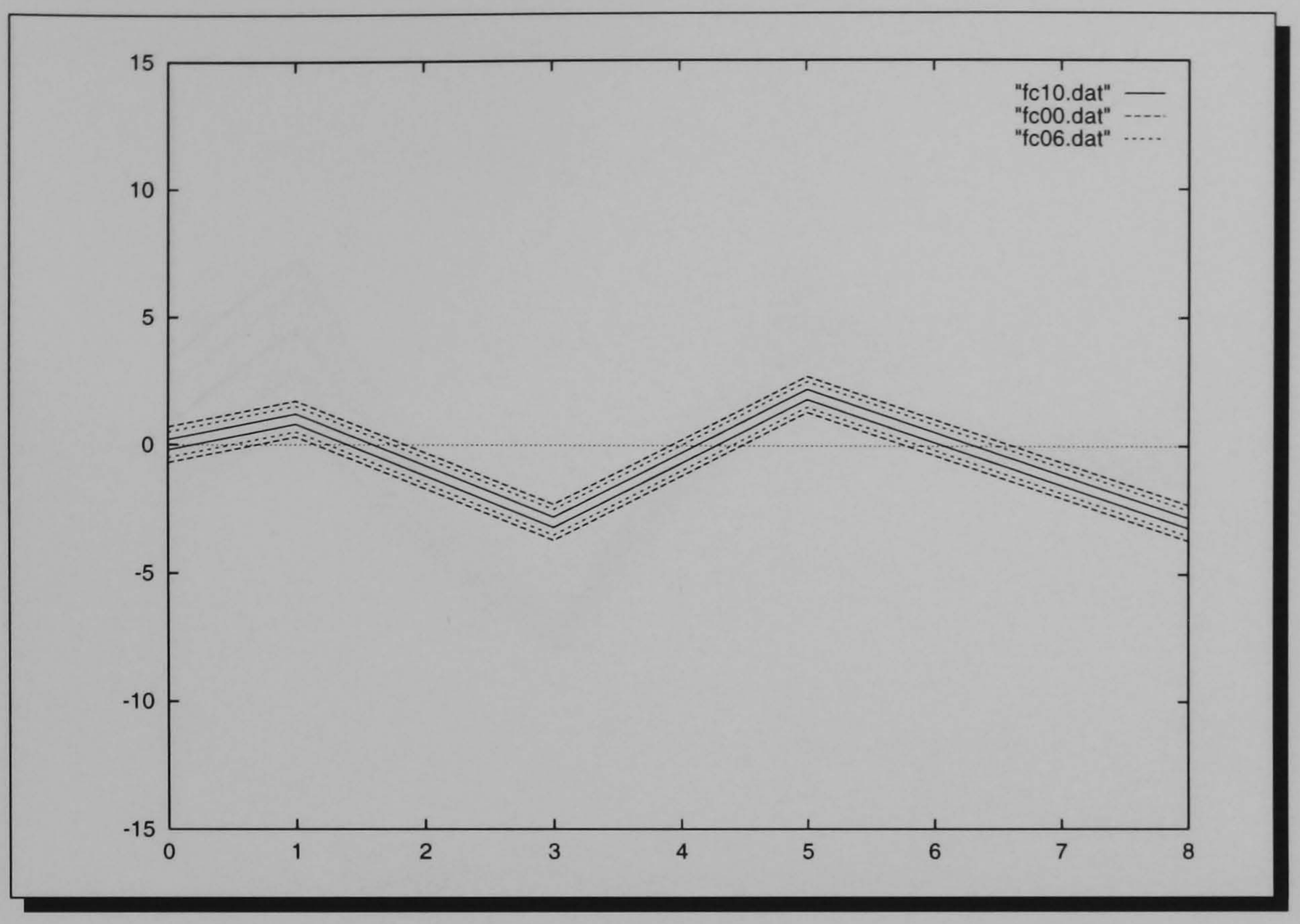


Figure 5.17: Strongly Positive Interactive Multiplication: Fuzzy Curve FC_1

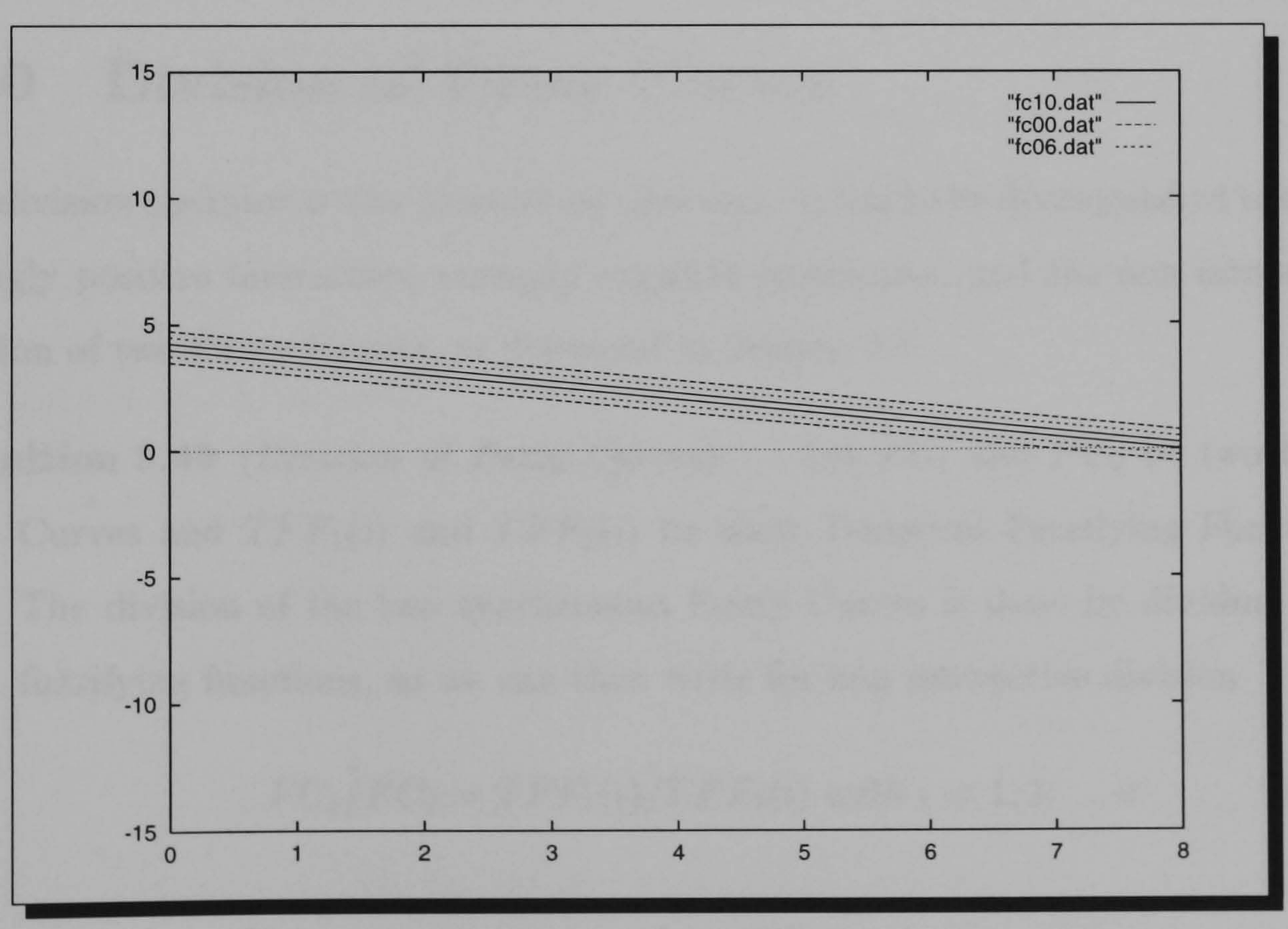


Figure 5.18: Strongly Positive Interactive Multiplication: Fuzzy Curve FC_2

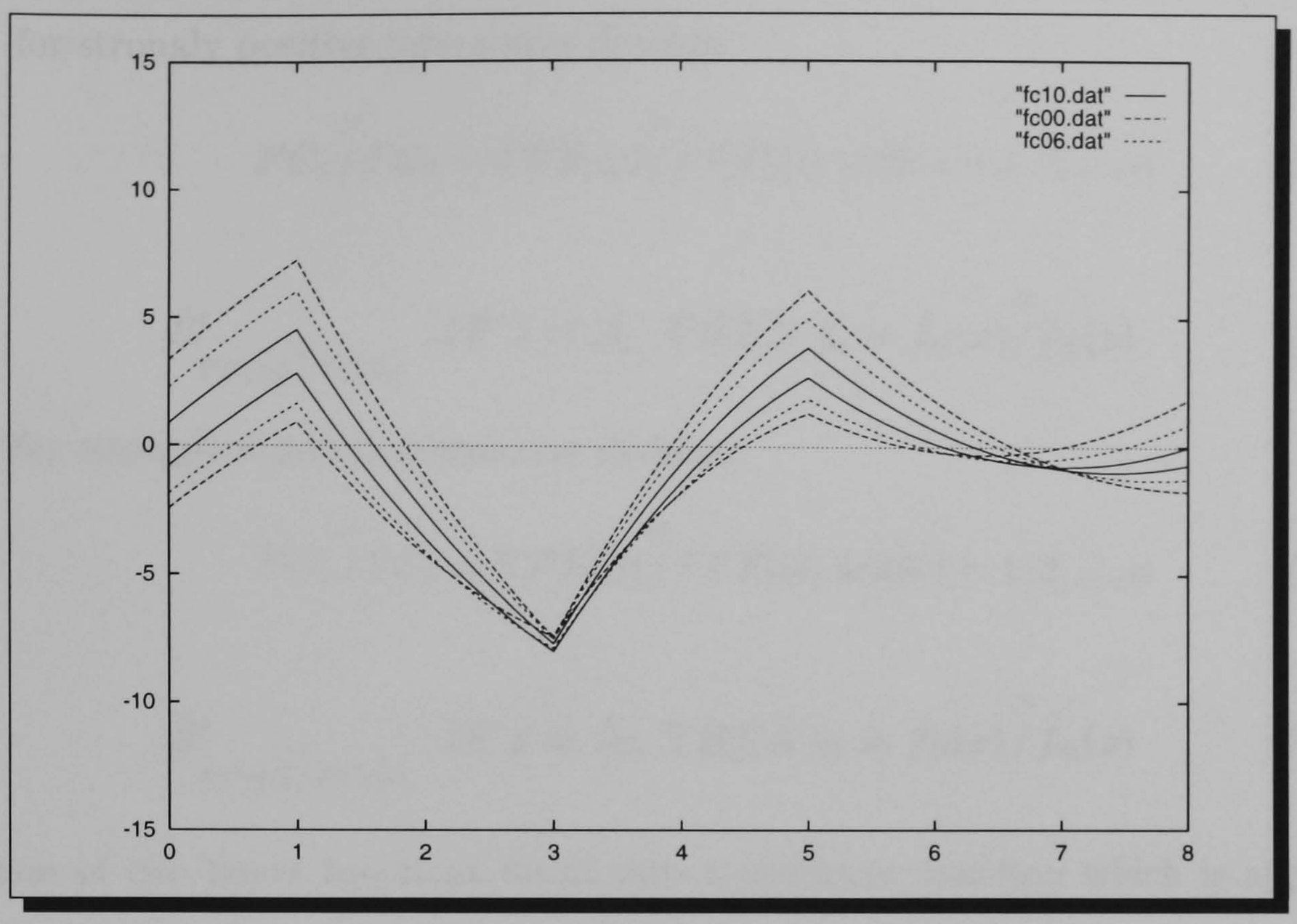


Figure 5.19: Strongly Positive Interactive Multiplication of FC_1 and FC_2

5.10 Division of Fuzzy Curves

The division operator is also a nonlinear operator. It has to be distinguished between strongly positive interactive, strongly negative interactive, and the non interactive division of two Fuzzy Curves, as discussed in Section 5.7.

Definition 5.49 (*Division of Fuzzy Curves*): Let FC_1 and FC_2 be two Fuzzy Curves and $TFF_1(i)$ and $TFF_2(i)$ be their Temporal Fuzzifying Functions. The division of the two synchronous Fuzzy Curves is done by dividing their fuzzifying functions, so we can then write for non interactive division

$$FC_1 \tilde{/} FC_2 = TFF_1(i) \tilde{/} TFF_2(i) \; with \; i = 1, 2, ..., n \tag{5.65}$$

$$\tilde{R}^i_{FC_1(i)\tilde{/}FC_2(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}_i \; is \; \tilde{f}_{1i}(x) \tilde{/} \tilde{f}_{2i}(x) \tag{5.66}$$

for strongly positive interactive division

$$FC_1 \overrightarrow{/} FC_2 = TFF_1(i) \overrightarrow{/} TFF_2(i) \ with \ i = 1, 2, ..., n \tag{5.67}$$

$$\tilde{R}^i_{FC_1(i) \overrightarrow{/} FC_2(i)} : IF \ x \ is \ \tilde{A}_i, \ THEN \ \tilde{y}_i \ is \ \tilde{f}_{1i}(x) \overrightarrow{/} \tilde{f}_{2i}(x) \tag{5.68}$$

for strongly negative interactive division

$$FC_1 \overleftrightarrow{/} FC_2 = TFF_1(i) \overleftrightarrow{/} TFF_2(i) \ with \ i = 1, 2, ..., n \tag{5.69}$$

$$\tilde{R}^i_{FC_1(i) \overleftrightarrow{/} FC_2(i)} : IF \ x \ is \ \tilde{A}_i, \ THEN \ \tilde{y}_i \ is \ \tilde{f}_{1i}(x) \overleftrightarrow{/} \tilde{f}_{2i}(x) \tag{5.70}$$

Division of two linear functions result into a quadratic function which is approximated by n number of linear functions as described in Section 5.9.

Definition 5.50 (*Non Interactive Division of Temporal Fuzzifying Functions*):

The non interactive division of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the division of their two membership functions non interactively, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \ with \ i = 1, 2, ..., n \tag{5.71}$$

then it can be written

$$\mu_{TFF_1 \tilde{/} TFF_2}(x, y) =$$

$$\begin{cases} 0 & if \quad y \leq min(TFF - Set_{>0.0}) \\ L(x, y) & if \quad min(TFF - Set_{>0.0}) < y < min(TFF - Set_{1.0}) \\ 1 & if \quad min(TFF - Set_{1.0}) \leq y \leq max(TFF - Set_{1.0}) \\ R(x, y) & if \quad max(TFF - Set_{1.0}) < y < max(TFF - Set_{>0.0}) \\ 0 & if \quad max(TFF - Set_{>0.0}) \leq y \end{cases} \tag{5.72}$$

with:

$$TFF - Set_{>0.0} = \{\frac{f1^{-}_{>0.0}(x)}{f2^{-}_{>0.0}(x)}, \frac{f1^{-}_{>0.0}(x)}{f2^{+}_{>0.0}(x)}, \frac{f1^{+}_{>0.0}(x)}{f2^{-}_{>0.0}(x)}, \frac{f1^{+}_{>0.0}(x)}{f2^{+}_{>0.0}(x)}\} \tag{5.73}$$

$$TFF - Set_{1.0} = \{\frac{f1^{-}_{1.0}(x)}{f2^{-}_{1.0}(x)}, \frac{f1^{-}_{1.0}(x)}{f2^{+}_{1.0}(x)}, \frac{f1^{+}_{1.0}(x)}{f2^{-}_{1.0}(x)}, \frac{f1^{+}_{1.0}(x)}{f2^{+}_{1.0}(x)}\} \tag{5.74}$$

$$L(x,y) = \frac{y - min(TFF - Set_{>0.0})}{min(TFF - Set_{1.0}) - min(TFF - Set_{>0.0})} \tag{5.75}$$

$$R(x,y) = \frac{max(TFF - Set_{>0.0}) - y}{max(TFF - Set_{>0.0}) - max(TFF - Set_{1.0})} \tag{5.76}$$

A non interactive division may not contain zero in the denominator.

5.10.1 Example: Non Interactive Division of Fuzzy Curves

Fig. 5.22 shows a non interactive division of the two Fuzzy Curves FC1 and FC2. There has been used a 3-point approximation. The Fuzzy Curves are defined by three α-cut levels:

α-cut level	data file
> 0.0	fc00.dat
0.6	fc06.dat
1.0	fc10.dat

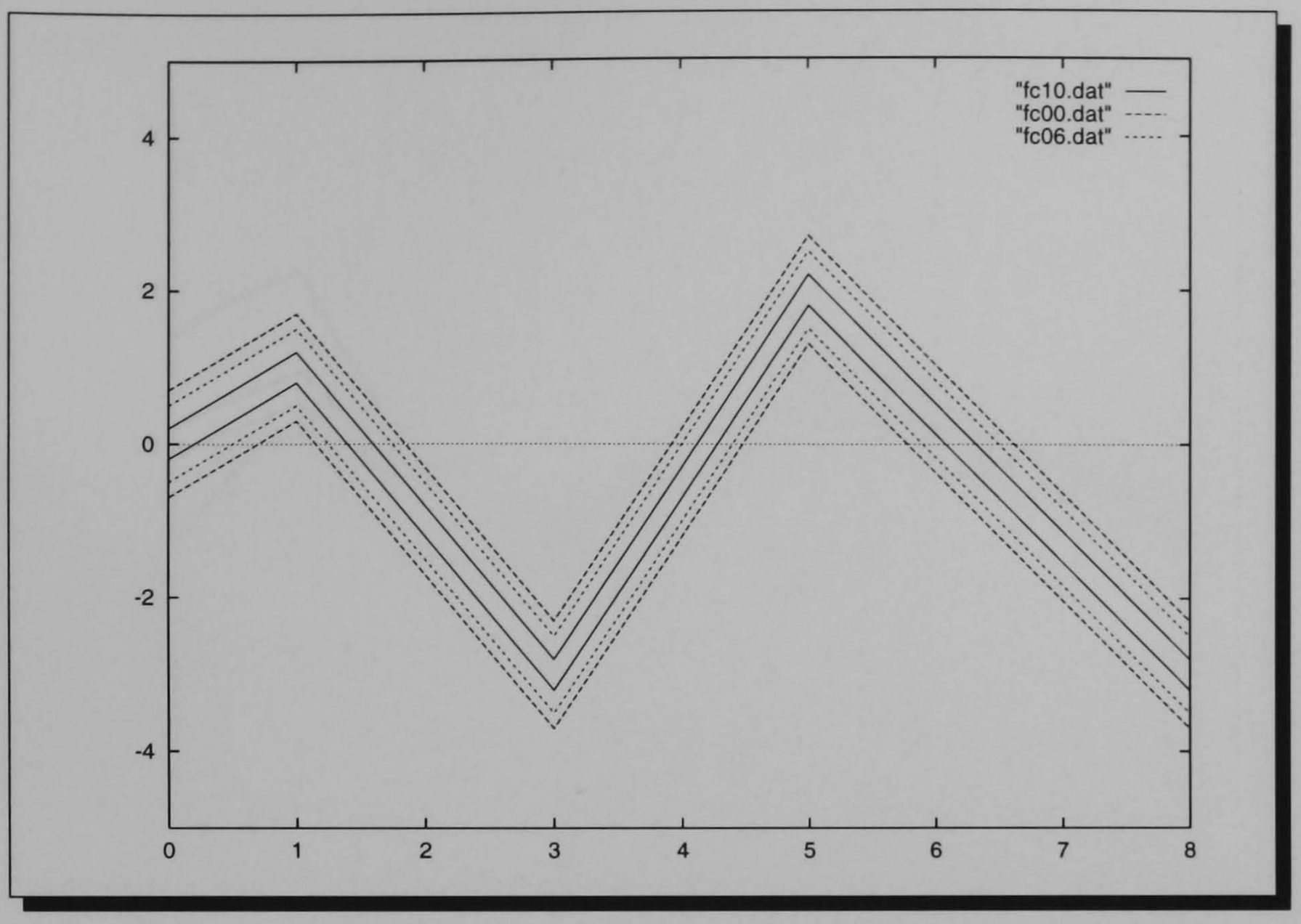


Figure 5.20: Non Interactive Division: Fuzzy Curve FC_1

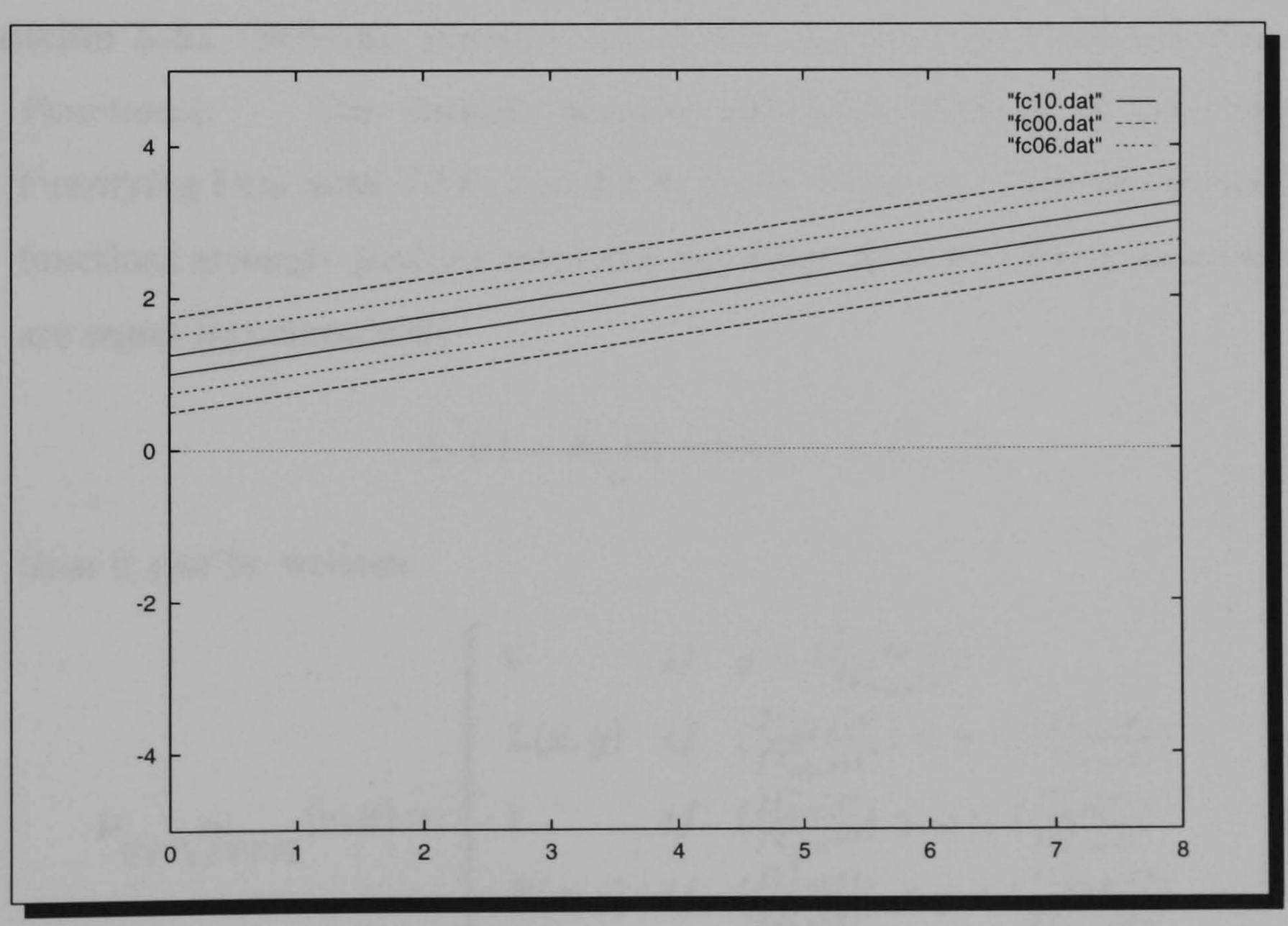


Figure 5.21: Non Interactive Division: Fuzzy Curve FC_2

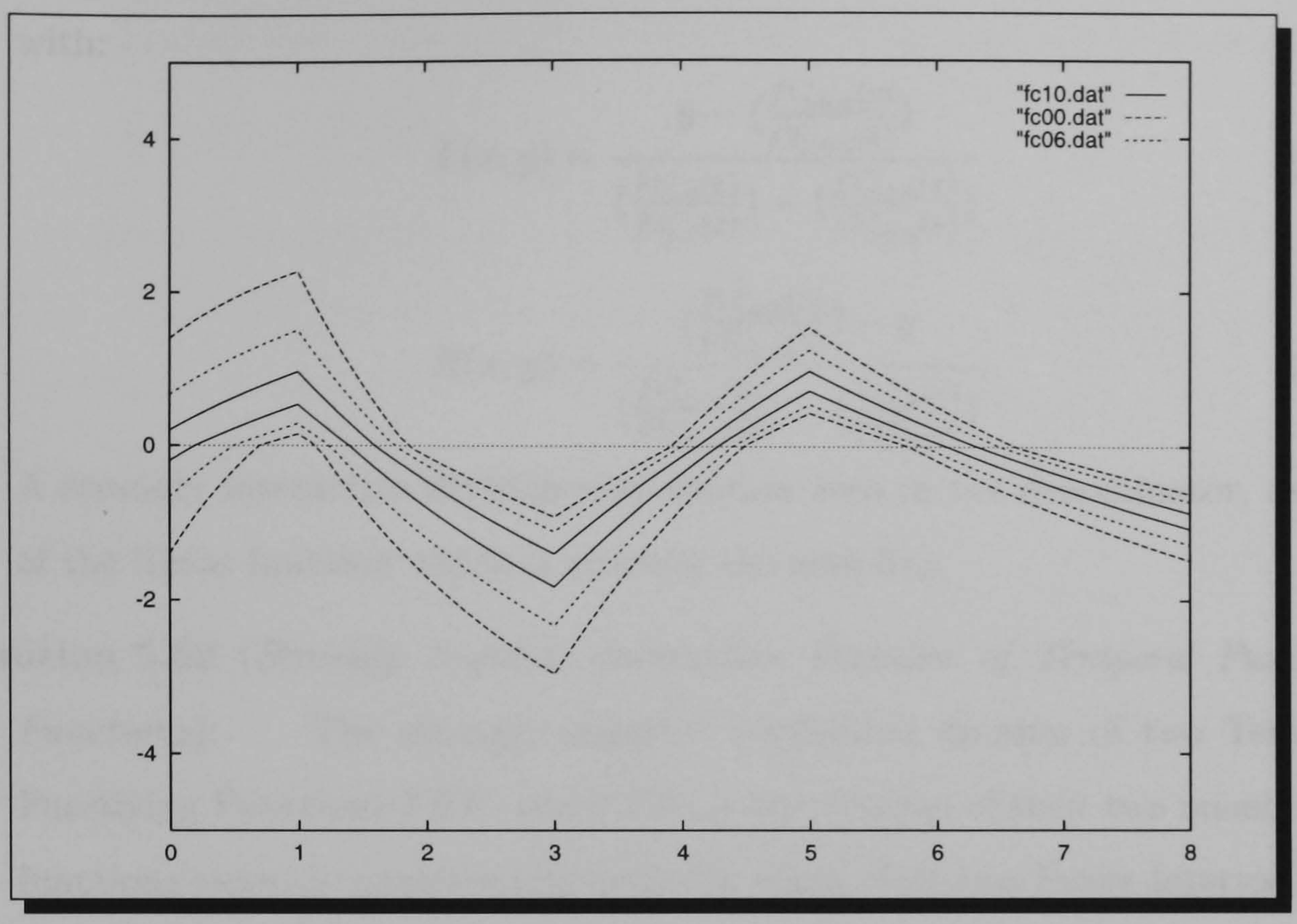


Figure 5.22: Non Interactive Division of FC_1 and FC_2

Definition 5.51 (*Strongly Positive Interactive Division of Temporal Fuzzifying Functions*): The strongly positive interactive division of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the division of their two membership functions strongly positive interactively, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \; with \; i = 1, 2, ..., n \tag{5.77}$$

then it can be written

$$\mu_{TFF_1 \overrightarrow{/} TFF_2}(x,y) = \begin{cases} 0 & if \quad y \leq (\frac{f1^-_{>0.0}(x)}{f2^-_{>0.0}(x)}) \\ L(x,y) & if \quad (\frac{f1^-_{>0.0}(x)}{f2^-_{>0.0}(x)}) < y < (\frac{f1^-_{1.0}(x)}{f2^-_{1.0}(x)}) \\ 1 & if \quad (\frac{f1^-_{1.0}(x)}{f2^-_{1.0}(x)}) \leq y \leq (\frac{f1^+_{1.0}(x)}{f2^+_{1.0}(x)}) \\ R(x,y) & if \quad (\frac{f1^+_{1.0}(x)}{f2^+_{1.0}(x)}) < y < (\frac{f1^+_{>0.0}(x)}{f2^+_{>0.0}(x)}) \\ 0 & if \quad (\frac{f1^+_{>0.0}(x)}{f2^+_{>0.0}(x)}) \leq y \end{cases} \tag{5.78}$$

with:

$$L(x,y) = \frac{y - (\frac{f1^{-}_{>0.0}(x)}{f2^{-}_{>0.0}(x)})}{(\frac{f1^{-}_{1.0}(x)}{f2^{-}_{1.0}(x)}) - (\frac{f1^{-}_{>0.0}(x)}{f2^{-}_{>0.0}(x)})} \tag{5.79}$$

$$R(x,y) = \frac{(\frac{f1^{+}_{>0.0}(x)}{f2^{+}_{>0.0}(x)}) - y}{(\frac{f1^{+}_{>0.0}(x)}{f2^{+}_{>0.0}(x)}) - (\frac{f1^{+}_{1.0}(x)}{f2^{+}_{1.0}(x)})} \tag{5.80}$$

A strongly interactive division may contain zero in the denominator, as part of the linear function which is crossing the zero line.

Definition 5.52 (*Strongly Negative Interactive Division of Temporal Fuzzifying Functions*): The strongly negative interactive division of two Temporal Fuzzifying Functions TFF_1 and TFF_2 is the division of their two membership functions strongly negative interactively, when their two Fuzzy Interval input are equal (synchronized)

$$\tilde{A_1}'(i) = \tilde{A_2}'(i) \; with \; i = 1, 2, ..., n \tag{5.81}$$

then it can be written

$$\mu_{TFF_1 \; \overrightarrow{/} \; TFF_2}(x,y) = \begin{cases} 0 & if \quad y \leq (\frac{f1^{-}_{>0.0}(x)}{f2^{+}_{>0.0}(x)}) \\ L(x,y) & if \quad (\frac{f1^{-}_{>0.0}(x)}{f2^{+}_{>0.0}(x)}) < y < (\frac{f1^{-}_{1.0}(x)}{f2^{+}_{1.0}(x)}) \\ 1 & if \quad (\frac{f1^{-}_{1.0}(x)}{f2^{+}_{1.0}(x)}) \leq y \leq (\frac{f1^{+}_{1.0}(x)}{f2^{-}_{1.0}(x)}) \\ R(x,y) & if \quad (\frac{f1^{+}_{1.0}(x)}{f2^{-}_{1.0}(x)}) < y < (\frac{f1^{+}_{>0.0}(x)}{f2^{-}_{>0.0}(x)}) \\ 0 & if \quad (\frac{f1^{+}_{>0.0}(x)}{f2^{-}_{>0.0}(x)}) \leq y \end{cases} \tag{5.82}$$

with:

$$L(x,y) = \frac{y - (\frac{f1^{-}_{>0.0}(x)}{f2^{+}_{>0.0}(x)})}{(\frac{f1^{-}_{1.0}(x)}{f2^{+}_{1.0}(x)}) - (\frac{f1^{-}_{>0.0}(x)}{f2^{+}_{>0.0}(x)})} \tag{5.83}$$

$$R(x,y) = \frac{(\frac{f1^{+}_{>0.0}(x)}{f2^{-}_{>0.0}(x)}) - y}{(\frac{f1^{+}_{>0.0}(x)}{f2^{-}_{>0.0}(x)}) - (\frac{f1^{+}_{1.0}(x)}{f2^{-}_{1.0}(x)})} \tag{5.84}$$

A strongly interactive division may contain zero in the denominator, as part of the linear function which is crossing the zero line.

5.10.2 Example: Strongly Positive Interactive Division of Fuzzy Curves

Fig. 5.25 shows a strongly positive interactive division of the two Fuzzy Curves FC1 and FC2. There has been used a 3-point approximation. The Fuzzy Curves are defined by three α-cut levels:

α-cut level	data file
> 0.0	fc00.dat
0.6	fc06.dat
1.0	fc10.dat

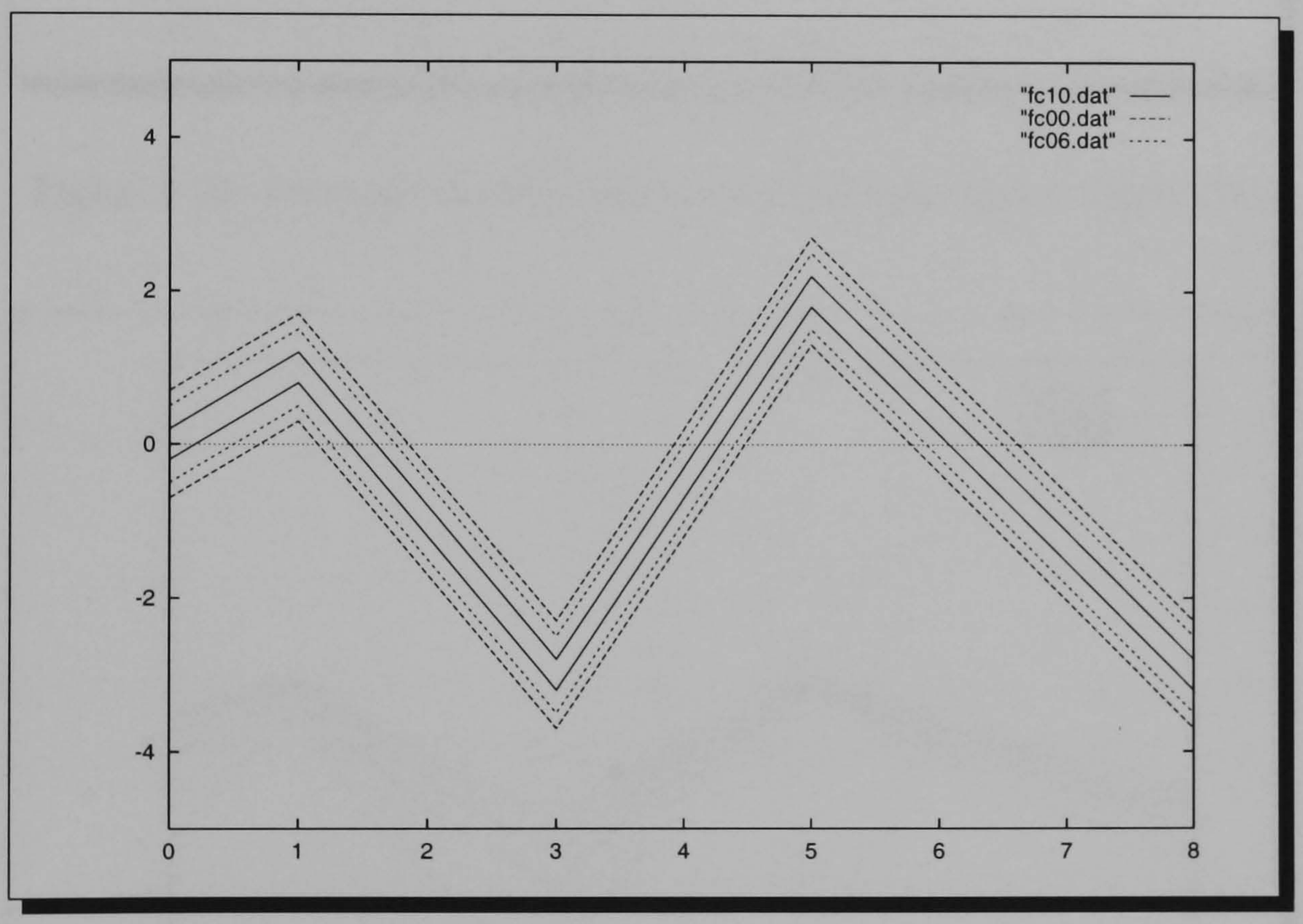


Figure 5.23: Strongly Positive Interactive Division: Fuzzy Curve FC_1

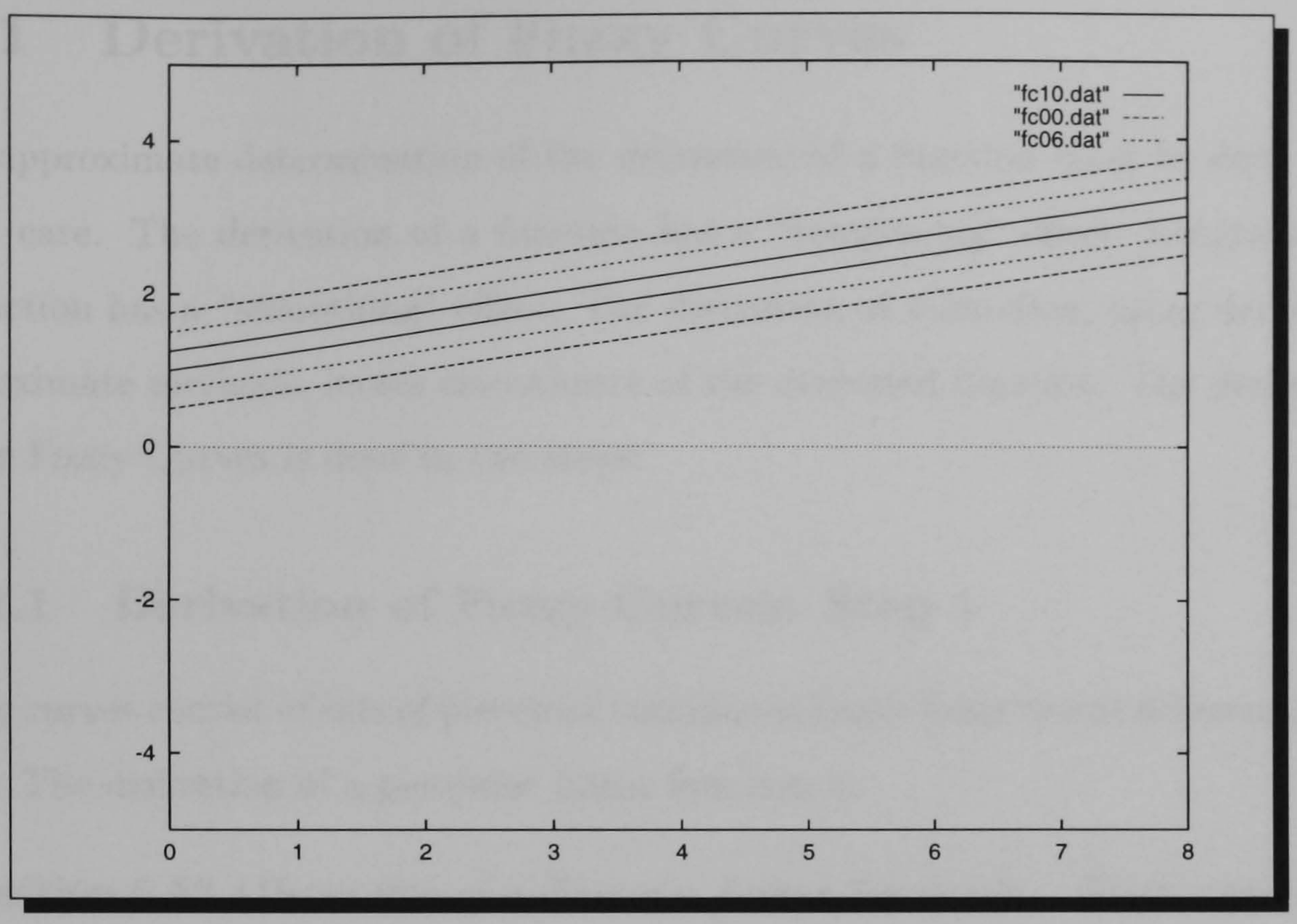


Figure 5.24: Strongly Positive Interactive Division: Fuzzy Curve FC_2

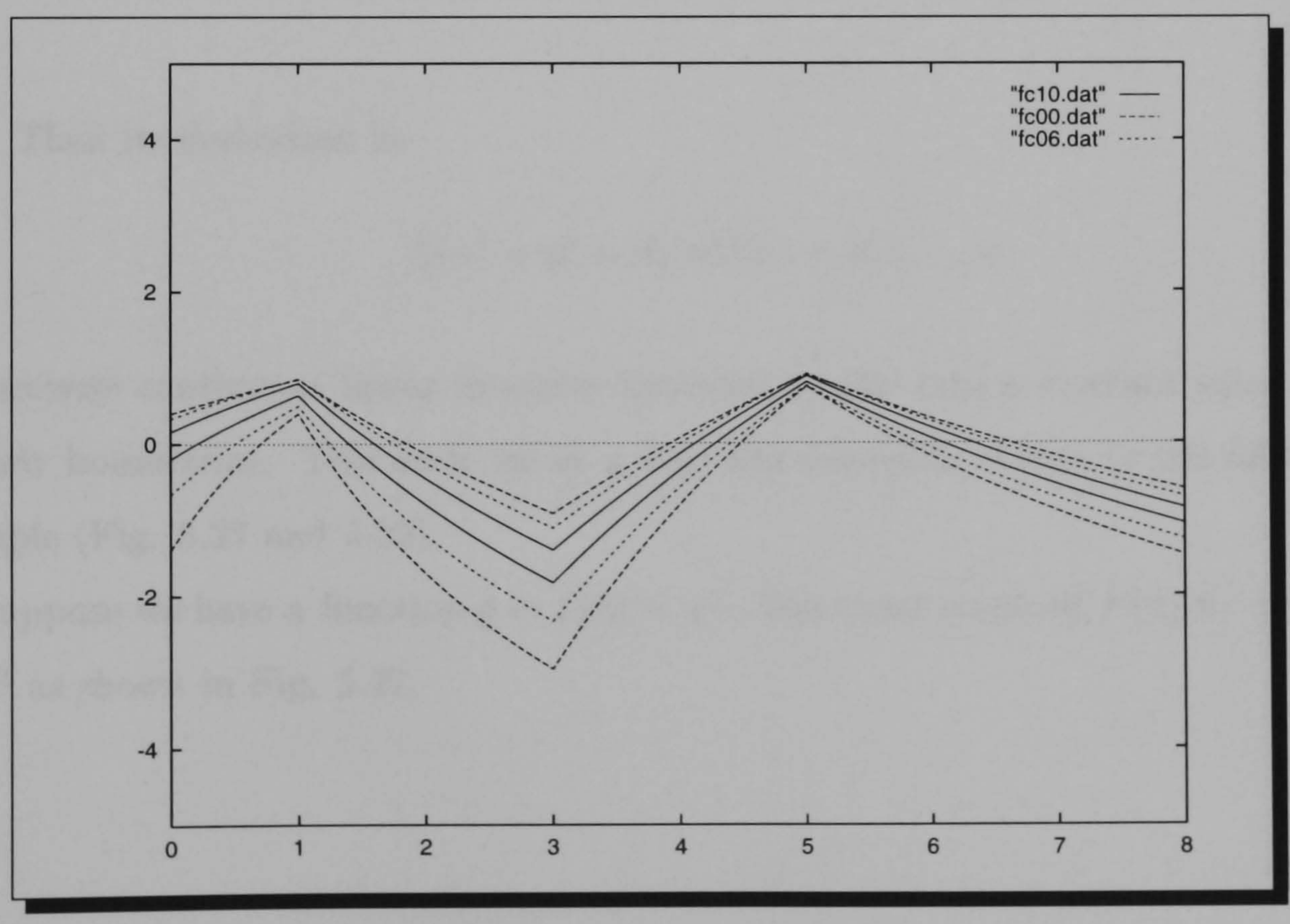


Figure 5.25: Strongly Positive Interactive Division of FC_1 and FC_2

5.11 Derivation of Fuzzy Curves

The approximate determination of the derivation of a function must be done with great care. The derivation of a function has a "roughening" effect; integration of a function has a "smoothing" effect. The derivation of a function, using derivative approximate methods, looses smoothness of the derivated function. The derivation of the Fuzzy Curves is done in two steps:

5.11.1 Derivation of Fuzzy Curves: Step 1

Fuzzy curves consist of sets of piecewise continuous linear functions at different α-cut level. The derivation of a piecewise linear function is:

Definition 5.53 (*Derivation of a Piecewise Linear Function*): Given a piecewise linear function with:

$$piecewise\ linear\ function: f_i(x) = y = A_i * x + B_i\ with\ i = 0, 1, ..., n \tag{5.85}$$

Then its derivation is:

$$f_i'(x) = y_i' = A_i\ with\ i = 0, 1, ..., n \tag{5.86}$$

A piecewise continuous linear function derivated results into a constant value valid in their boundaries. This ends up in a step like function, shown in the following example (Fig. 5.27 and 5.29).

Suppose we have a function $y = f(x) = x^3$. The exact result of $f'(x)$ is: $f'(x) = 3 * x^2$ as shown in Fig. 5.27.

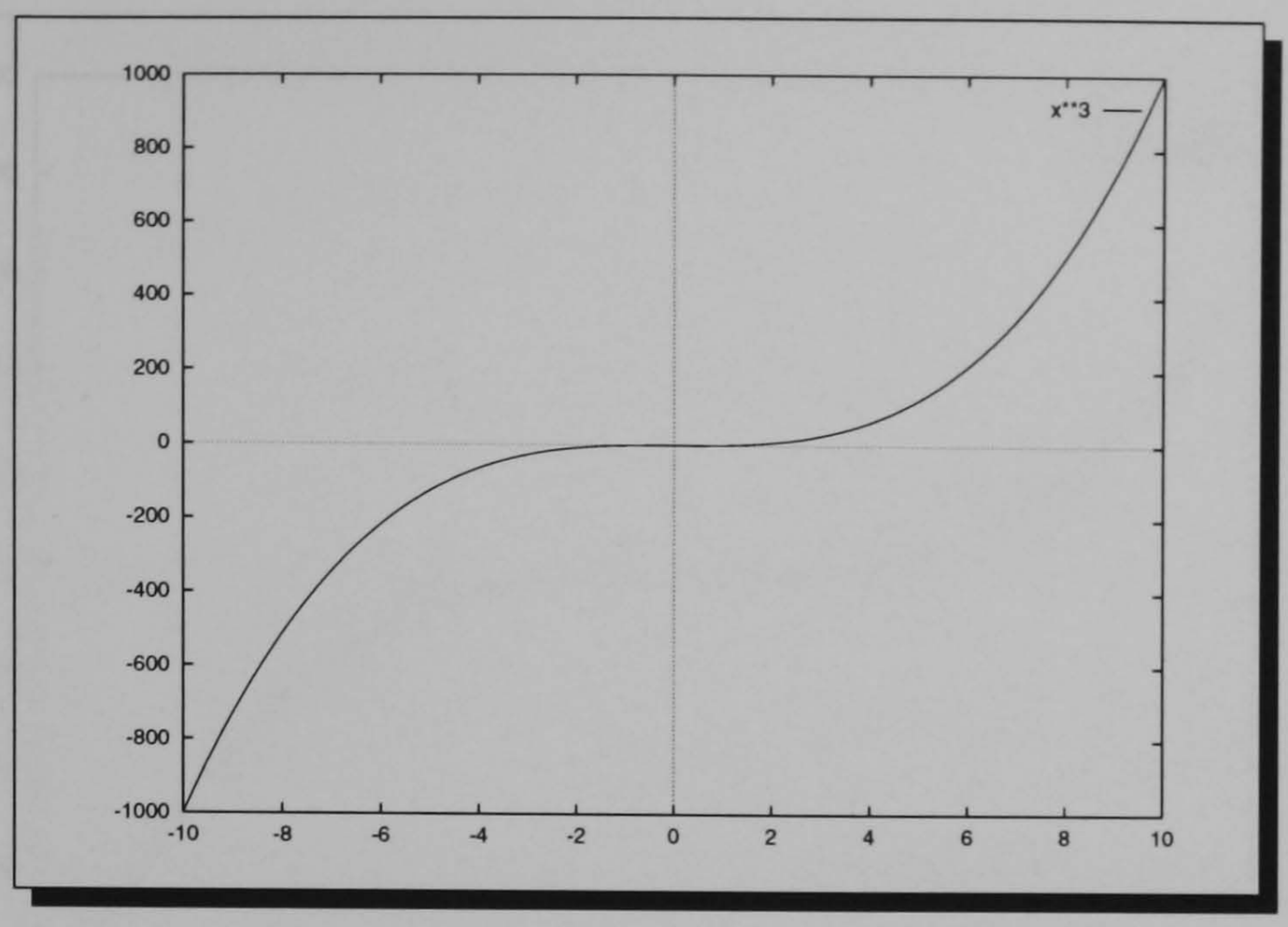


Figure 5.26: Function $f(x) = x^3$

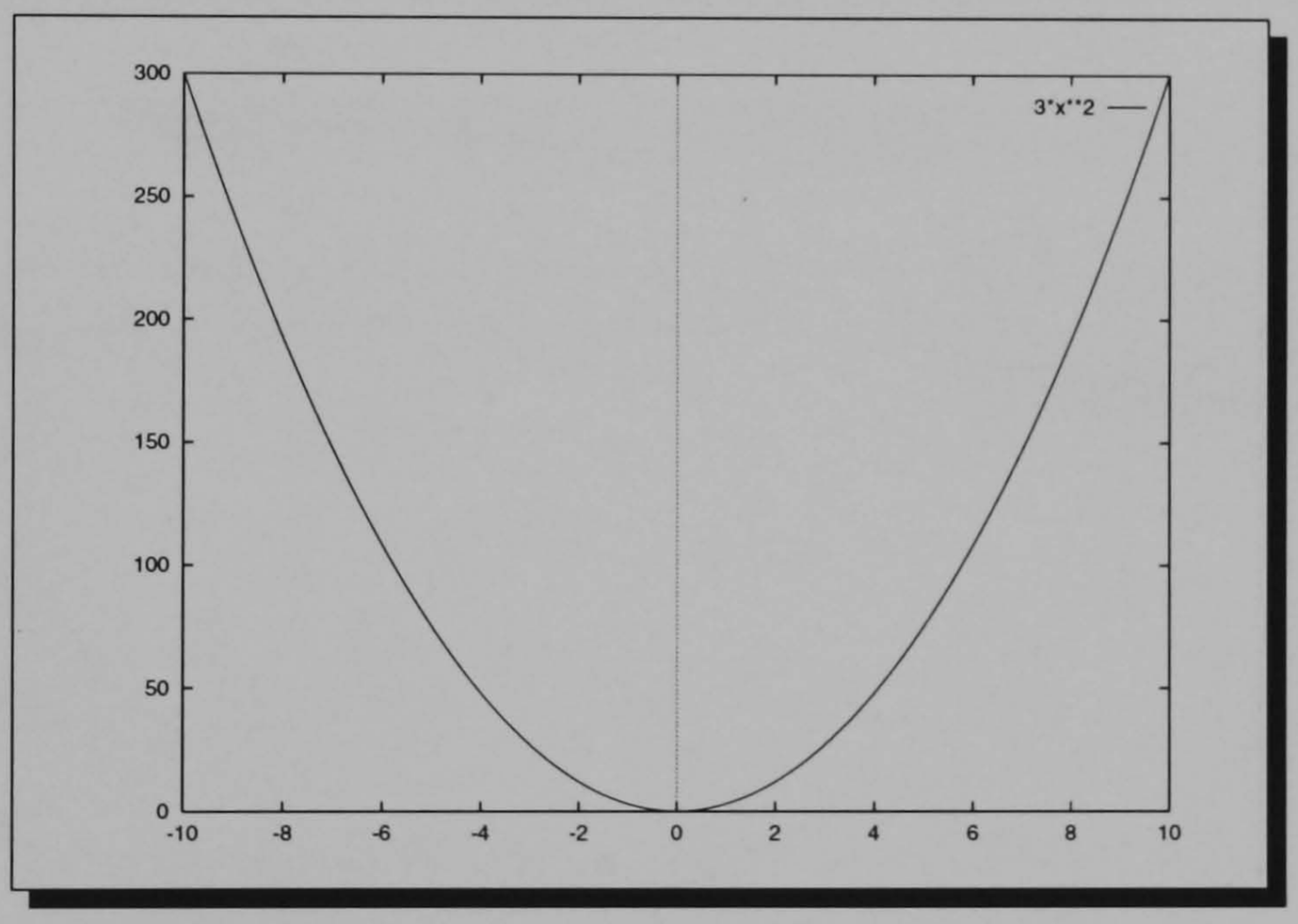


Figure 5.27: Exact Derivation of Function $f(x) : f'(x) = 3 * x^2$

Suppose we approximate the function $y = f(x) = x^3$ by 10 (20) linear functions. After derivation we get a 10 (20) step functions as shown in Fig. 5.29.

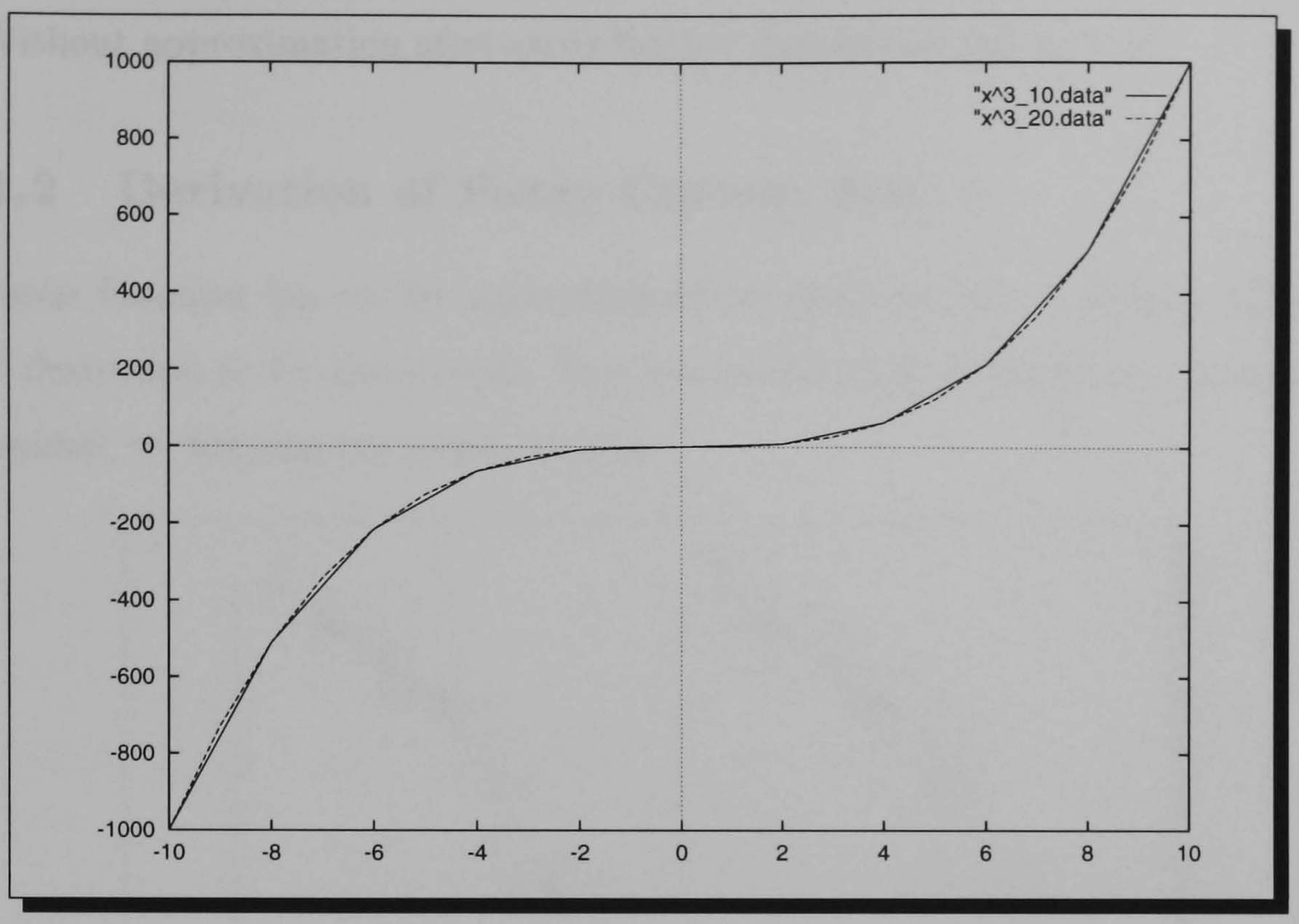


Figure 5.28: Approximate Function: $f(x) = x^3$

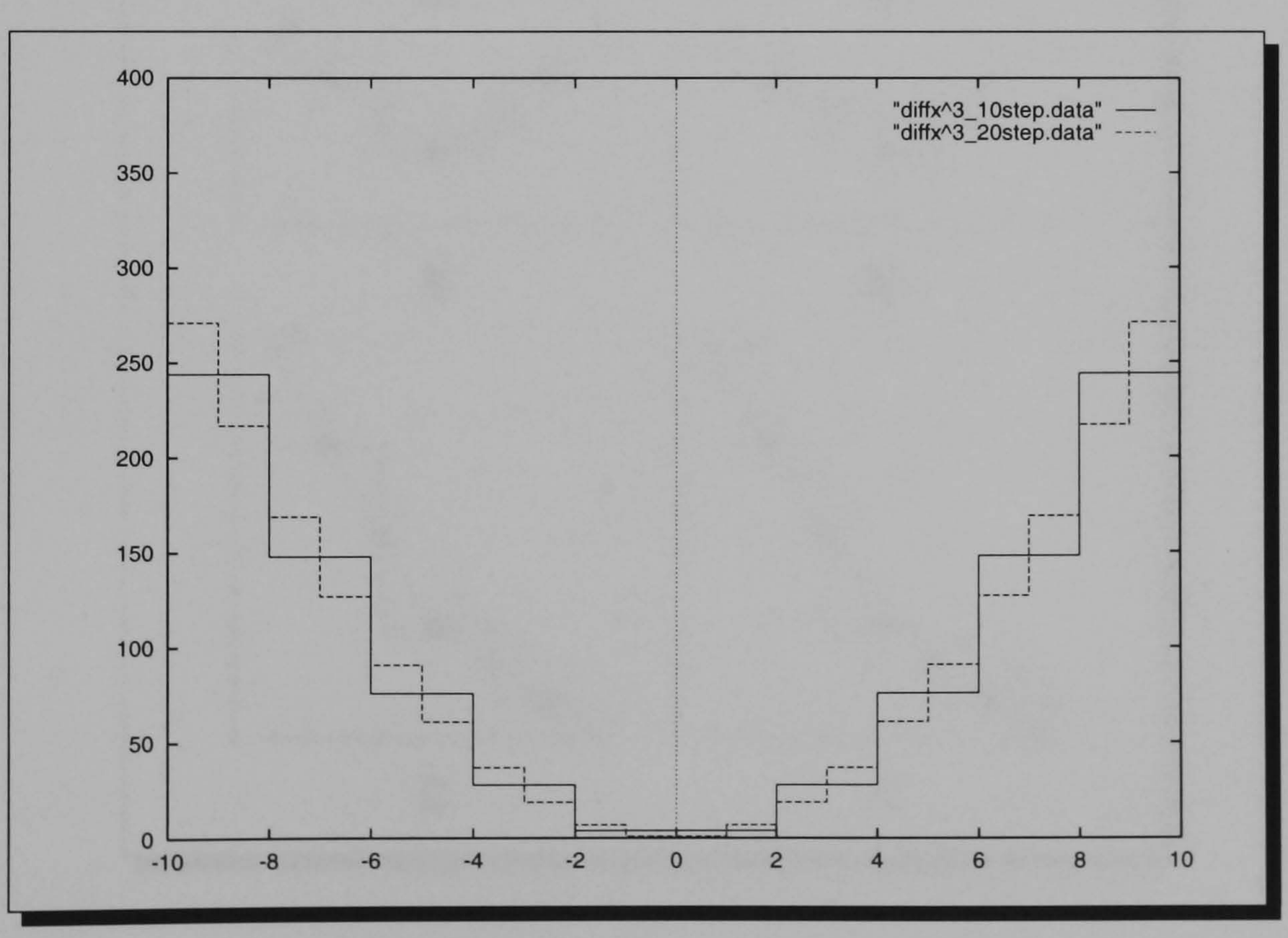


Figure 5.29: Approximate Derivation of Function $f(x)$: $f'(x) =$ step function

Without approximation afterwards further derivations will be null.

5.11.2 Derivation of Fuzzy Curves: Step 2

The step function has to be approximated to eliminate the drawback of second order derivation to be always null. New synchronization of the Fuzzy Curves must be avoided, by keeping the origin x-values.

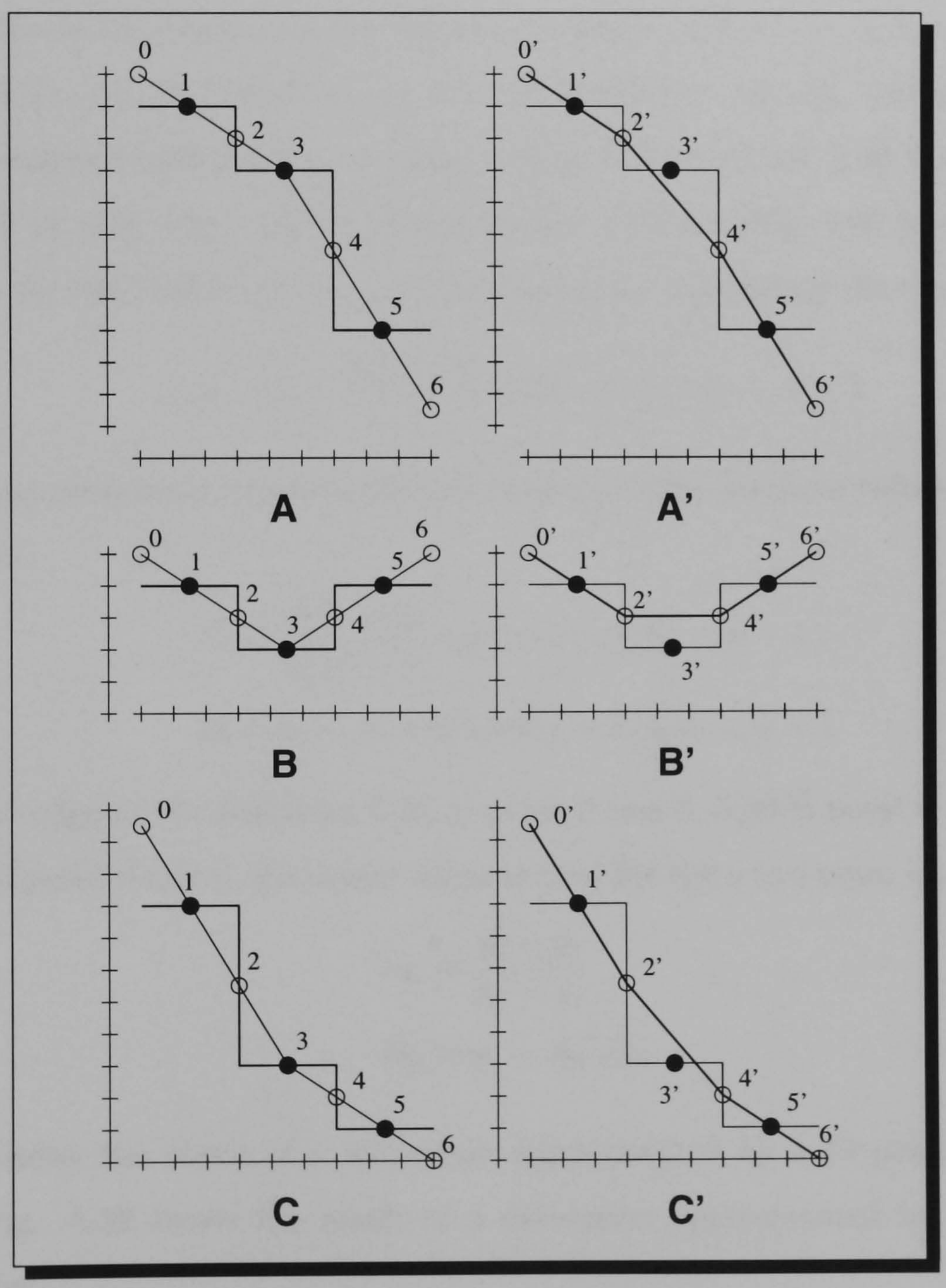


Figure 5.30: Calculating Y-Values Given Different Step Function Shapes

The Fig. 5.30 A, 5.30 B, and 5.30 C show how the y-values are calculated, given different kind of step function shapes.

Definition 5.54 (*Approximation of Step Function*): The x-values for the y-values in the center (Fig. 5.30A x1, x3, x5, 5.30B x1, x3, x5, and 5.30C x1, x3, x5):

$$x_i = x_i + \frac{x_{i+1} - x_i}{2} \; with \; i = 1, 3, 5, ..., n-1 \tag{5.87}$$

Y-values in the center are the derivation values A_i with $i = 1, 2, 3, ..., n$ (Fig. 5.30 A y1, y3, y5, 5.30 B y1, y3, y5, and 5.30 C y1, y3, y5). The x-values for the y-values which must be calculated (Fig. 5.30 A x2, x4, 5.30 B x2, x4, and 5.30 C x2, x4): The y-values at this specific x-values (Fig. 5.30 A y2, y4, 5.30 B y2, y4, and 5.30 C y2, y4) are determined by calculating the mean values:

$$y_{i+1} = y_i - \frac{y_{i+2} - y_i}{2} \; with \; i = 1, 3, 5, ..., n-1 \tag{5.88}$$

The approximation is a two point interpolation with the mean values calculated before:

$$\begin{aligned} A_i &= \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \; with \; i = 2, 4, 6, ..., n-1 \\ B_i &= y_i - A_i * x_i \; with \; i = 2, 4, 6, ..., n-1 \end{aligned} \tag{5.89}$$

At the edge of the functions 5.30 A point 0 and 6, 5.30 B point 0 and 6, and 5.30 C point 0 and 6, the center value is used for the a two point interpolation:

$$\begin{aligned} A_0 &= \frac{y_2 - y_1}{x_2 - x_1} \\ B_0 &= y_1 - A_0 * x_1 \end{aligned} \tag{5.90}$$

Fig. 5.31 shows the result of a derivation approximated by a 10 point interpolation and Fig. 5.32 shows the result of a derivation approximated by a 20 point interpolation.

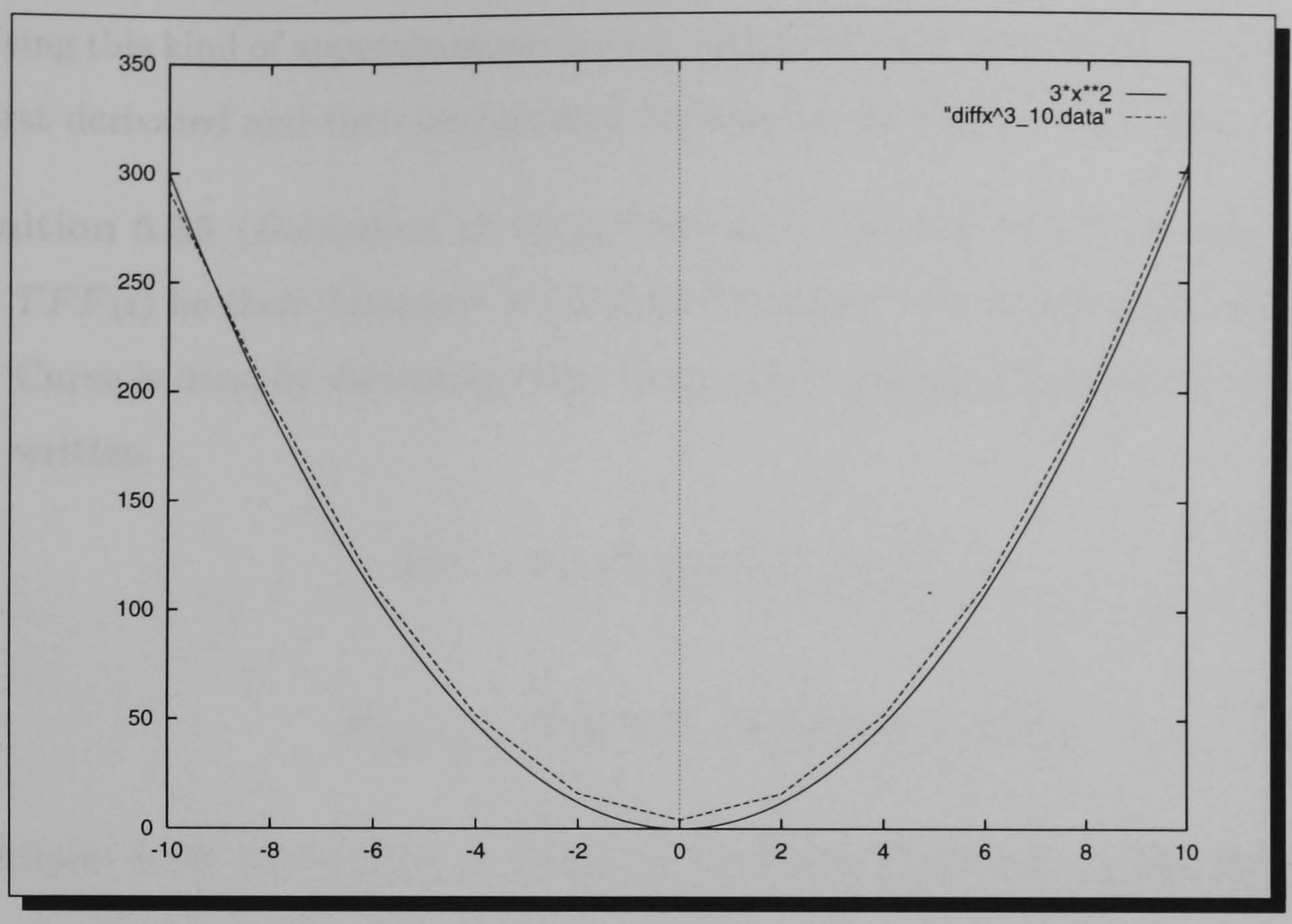


Figure 5.31: Approximate Derivation With 10 Point Interpolation

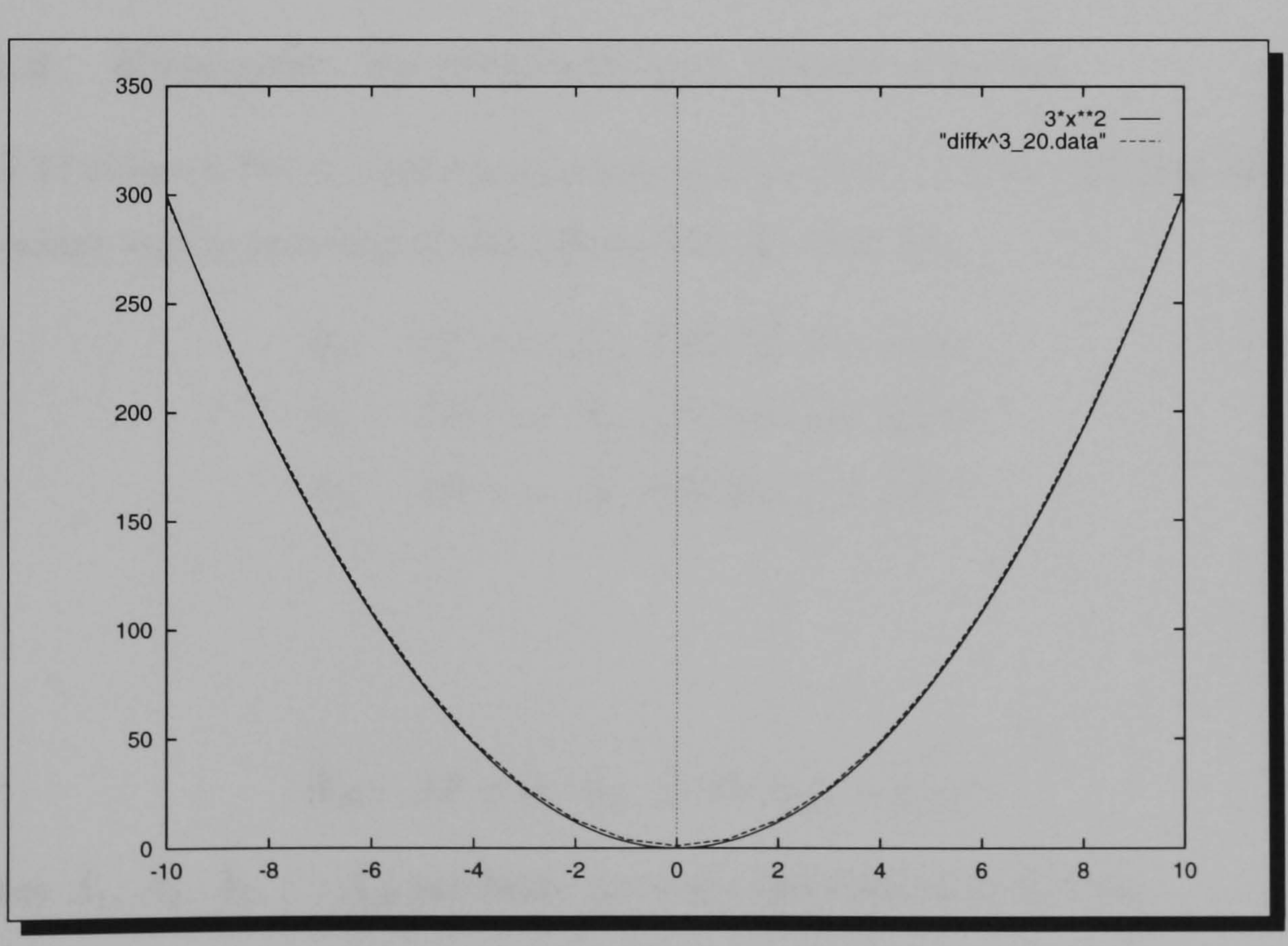


Figure 5.32: Approximate Derivation With 20 Point Interpolation

Using this kind of approximation for derivation of Fuzzy Curves the Fuzzy Curves are first derivated and then interpolated to build a new Fuzzy Curve again.

Definition 5.55 (*Derivation of Fuzzy Curves*): Let FC be a Fuzzy Curve and $TFF(i)$ be their Temporal Fuzzifying Functions. The derivation of the Fuzzy Curve is done by derivating their Temporal Fuzzifying Functions, so it can be written

$$FC' = TFF'(i) \; with \; i = 1, 2, ..., n \tag{5.91}$$

$$\tilde{R}^i_{TFF'(i)} : IF \; x \; is \; \tilde{A}_i, \; THEN \; \tilde{y}'_i \; is \; \tilde{f}'_i(x) \tag{5.92}$$

Definition 5.56 (*Derivation of Temporal Fuzzifying Function*): The derivation of a Temporal Fuzzifying Function TFF is the derivation of each α-cut-level function which is done in 2 steps as described in 5.11.1 and 5.11.2.

5.11.3 Example: Derivation of a Fuzzy Curve

Fig. 5.33 shows a Fuzzy Curve approximating the $f(x) = x^3$ by 20 linear functions (a 21-point approximation) at the α-level 0.0, 0.6, and 1.0.

$$\begin{aligned}
\tilde{R}_1&: \quad IF \; x \; is \; \tilde{A}_1, \; THEN \; \tilde{y} \; is \; \tilde{f}_1(x) \\
\tilde{R}_2&: \quad IF \; x \; is \; \tilde{A}_2, \; THEN \; \tilde{y} \; is \; \tilde{f}_2(x) \\
\tilde{R}_3&: \quad IF \; x \; is \; \tilde{A}_3, \; THEN \; \tilde{y} \; is \; \tilde{f}_3(x) \\
&\cdot \qquad \cdot \\
&\cdot \qquad \cdot \\
&\cdot \qquad \cdot \\
\tilde{R}_{20}&: \quad IF \; x \; is \; \tilde{A}_{20}, \; THEN \; \tilde{y} \; is \; \tilde{f}_{20}(x)
\end{aligned}$$

whereas $\tilde{A}_1$, $\tilde{A}_2$, $\tilde{A}_3$, ..., $\tilde{A}_{20}$ are fuzzy intervals and defined as follows:

$$\begin{aligned}
\tilde{A}_1 &= (-10, -9, 0.1, 0.1) \\
\tilde{A}_2 &= (-9, -8, 0.1, 0.1) \\
\tilde{A}_3 &= (-8, -7, 0.1, 0.1) \\
&\cdot \quad \cdot \\
&\cdot \quad \cdot \\
\tilde{A}_{20} &= (9, 10, 0.1, 0.1)
\end{aligned}$$

$\tilde{f}_1(x)$, $\tilde{f}_2(x)$, $\tilde{f}_3(x)$, ..., and $\tilde{f}_4(x)$ are linearly defined fuzzifying function and defined with the following α-level functions:

$\tilde{f}_1(x)$:

$$f^-_{>0.0}(x) = 271 * x + 1650 \quad f^+_{>0.0}(x) = 271 * x + 1770$$

$$f^-_{0.6}(x) = 271 * x + 1670 \quad f^+_{0.6}(x) = 271 * x + 1750$$

$$f^-_{1.0}(x) = 271 * x + 1690 \quad f^+_{1.0}(x) = 271 * x + 1730$$

$\tilde{f}_2(x)$:

$$f^-_{>0.0}(x) = 217 * x + 1164 \quad f^+_{>0.0}(x) = 217 * x + 1284$$

$$f^-_{0.6}(x) = 217 * x + 1184 \quad f^+_{0.6}(x) = 217 * x + 1264$$

$$f^-_{1.0}(x) = 217 * x + 1204 \quad f^+_{1.0}(x) = 217 * x + 1244$$

$\tilde{f}_3(x)$:

$$f^-_{>0.0}(x) = 169 * x + 780 \quad f^+_{>0.0}(x) = 169 * x + 900$$

$$f^-_{0.6}(x) = 169 * x + 800 \quad f^+_{0.6}(x) = 169 * x + 880$$

$$f^-_{1.0}(x) = 169 * x + 820 \quad f^+_{1.0}(x) = 169 * x + 860$$

.

.

.

$\tilde{f}_{20}(x)$:

$$f^{-}_{>0.0}(x) = 271 * x - 1770 \quad f^{+}_{>0.0}(x) = 271 * x - 1650$$

$$f^{-}_{0.6}(x) = 271 * x - 1750 \qquad f^{+}_{0.6}(x) = 271 * x - 1670$$

$$f^{-}_{1.0}(x) = 271 * x - 1730 \qquad f^{+}_{1.0}(x) = 271 * x - 1690$$

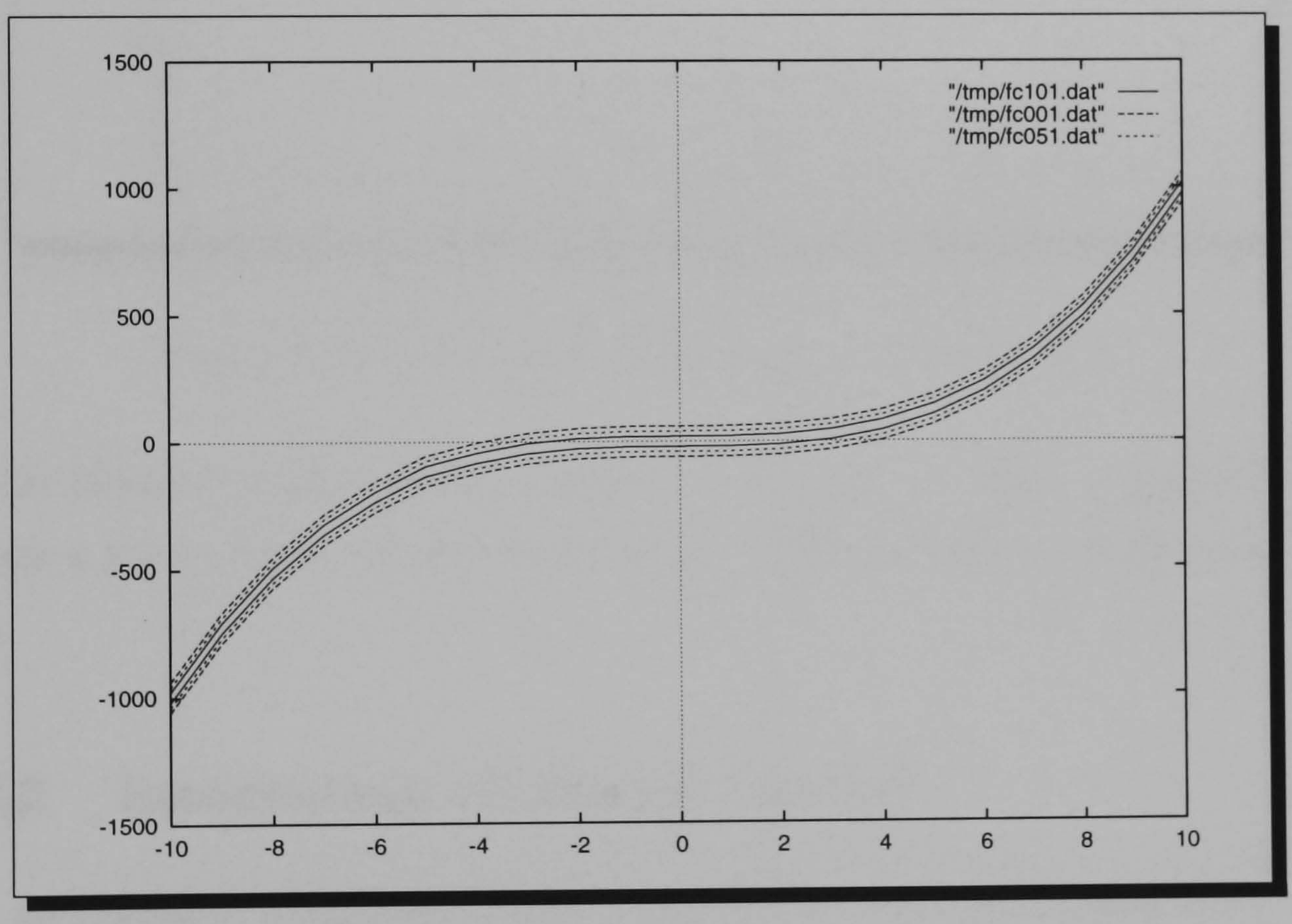


Figure 5.33: Fuzzy Curve Representing x^3

Fig. 5.34 represents the derivation of the Fuzzy Curve (Fig. 5.33).

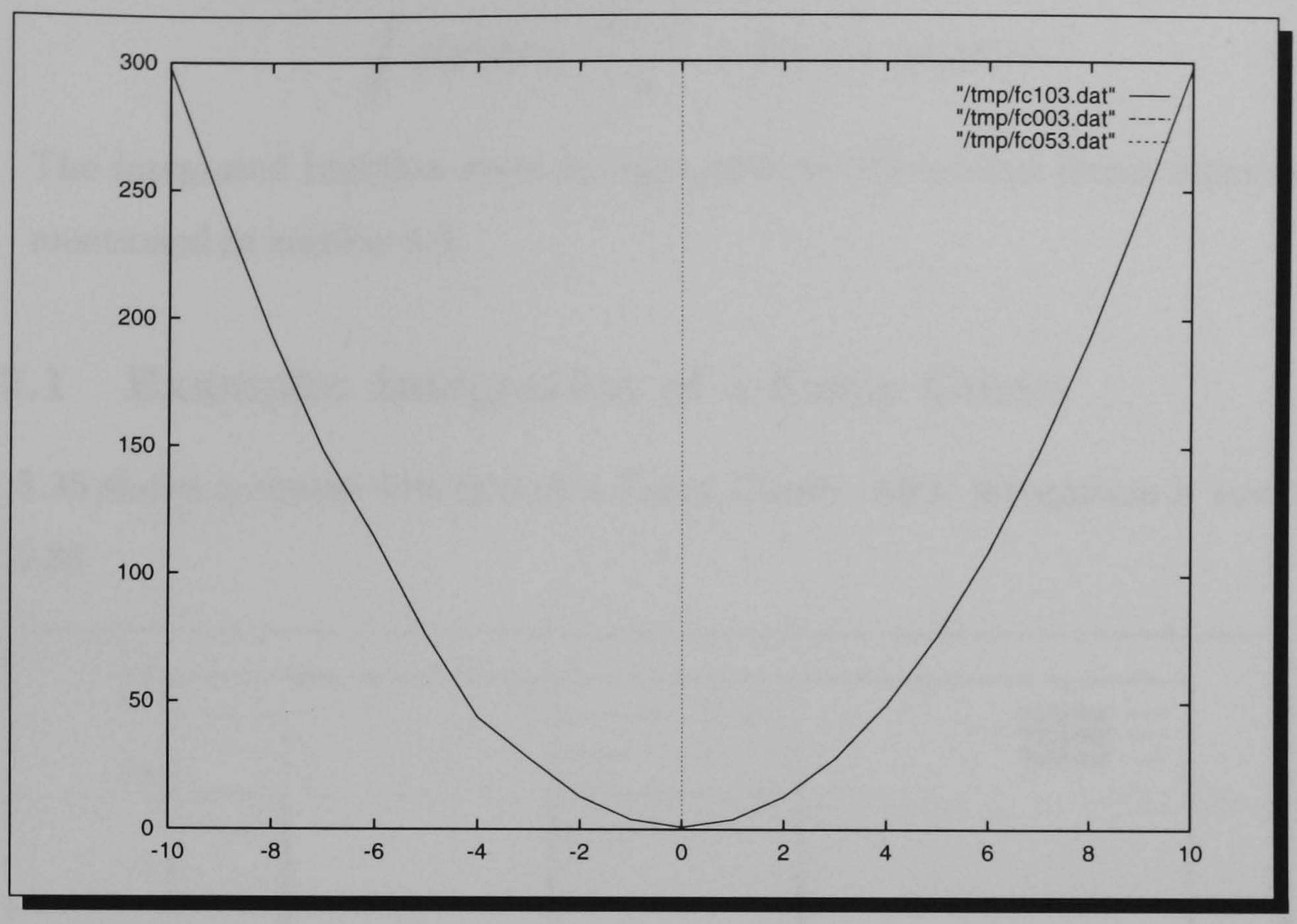


Figure 5.34: Fuzzy Curve Representing Derivation of x^3

The tolerance of the different α-levels is at any time the same. After derivation of such a Fuzzy Curve a fuzzy curve with no tolerances appears, as shown in Fig. 5.34.

5.12 Integration of Fuzzy Curves

The integration of a function of order n results into a function of order $n+1$.

Definition 5.57 (*Integration of Fuzzy Curve*): Fuzzy Curves are integrated by integrating each Temporal Fuzzifying Function.

$$\forall \alpha - cut - level\ with\ \alpha \in [0,1] : \tilde{f}_i(x) : f_\alpha(x) = A * x + B \tag{5.93}$$

is integrated:

$$\int f(x)dx = \frac{A * x^2}{2} + B * x + constant \tag{5.94}$$

The integrated function must be approximated by several linear functions as mentioned in section 5.9.

5.12.1 Example: Integration of a Fuzzy Curve

Fig. 5.35 shows a square function of a Fuzzy Curve. After integration it results in Fig. 5.36.

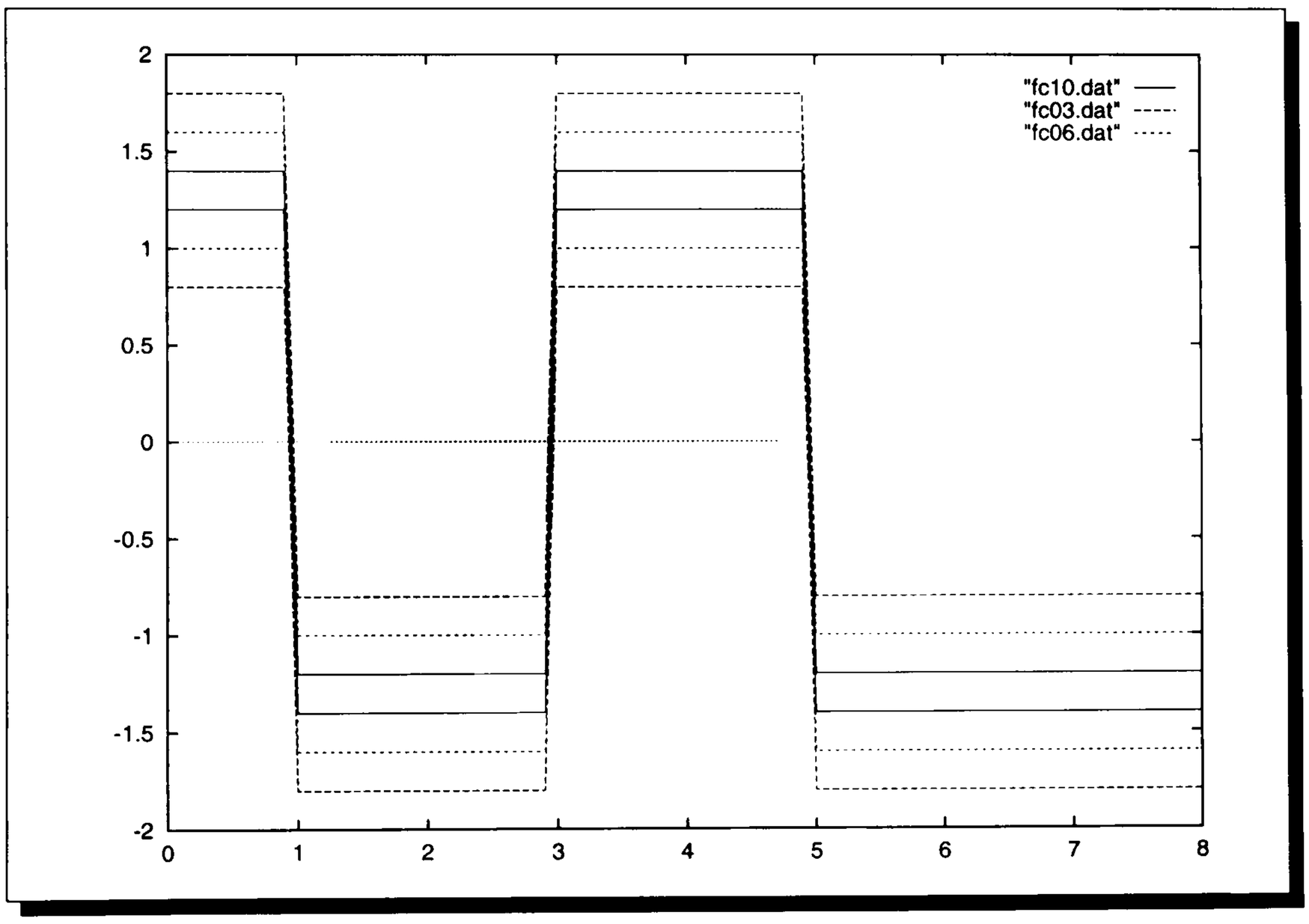


Figure 5.35: Fuzzy Curve Representing Square Function

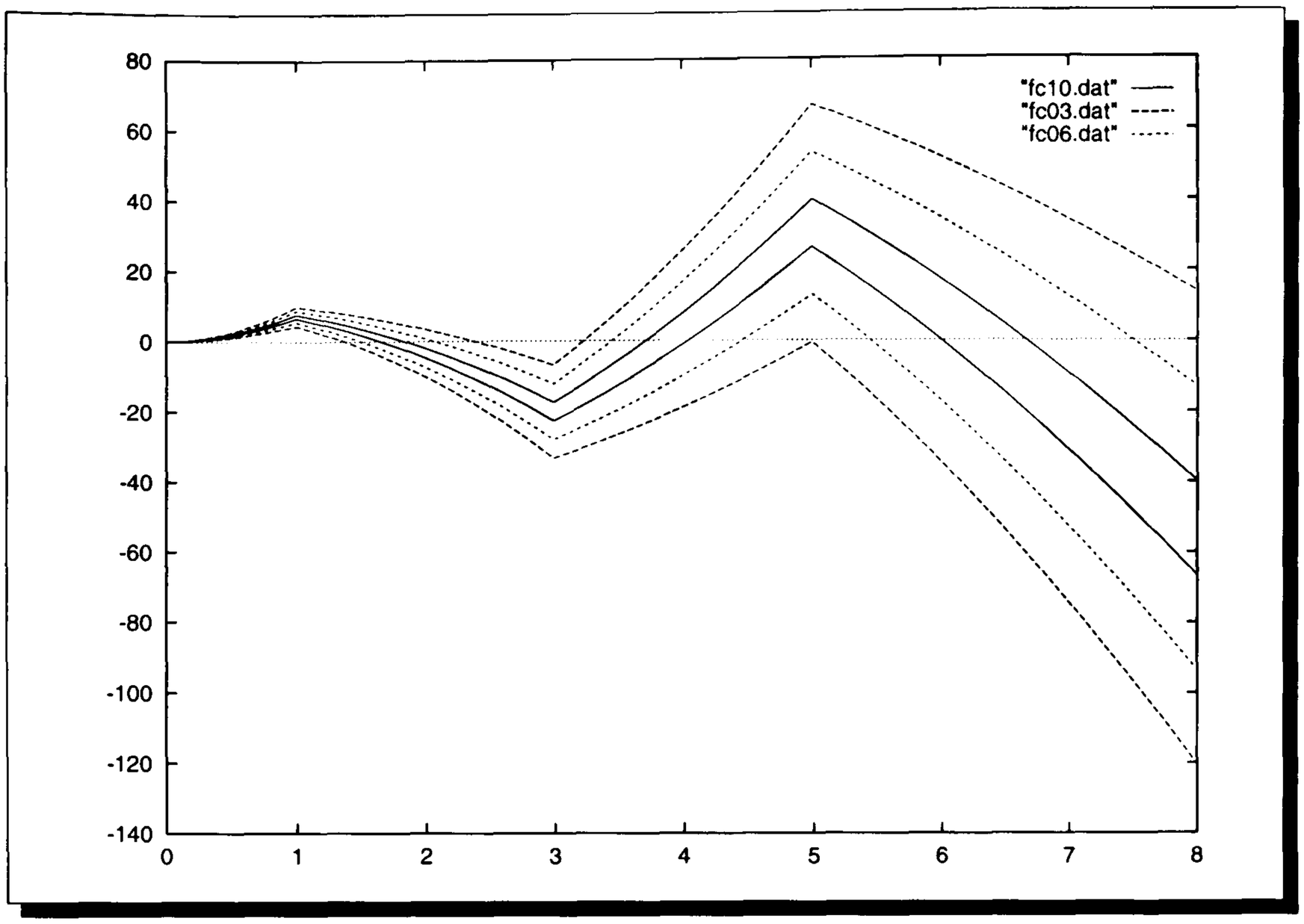


Figure 5.36: Fuzzy Curve Representing Integration of Square Function

5.13 Example: Kettle

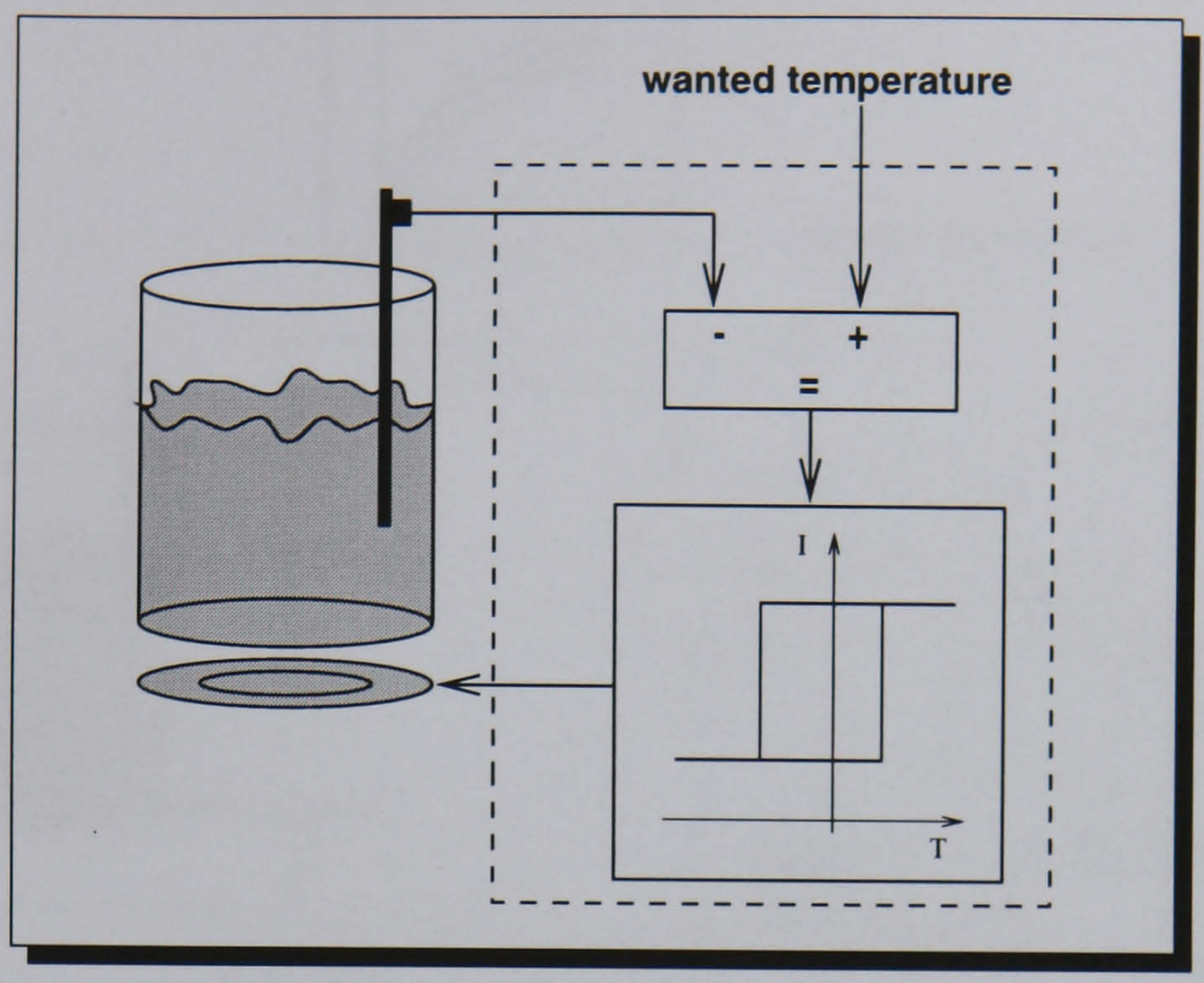


Figure 5.37: Simple Kettle Control

Fig. 5.37 shows a simple kettle control that switches on the heating system if the difference between `wanted temperature` and water temperature is $> 5^oC$. It switches off the heating system if the difference between `wanted temperature` and water temperature is $< 5^oC$. The hysteresis is wanted to avoid fluttering. This kettle works fine for most of the time. One problem is that some wanted temperatures are hard to adjust. This is because of the nonlinearity of the sensor and the heating system. The task is now to improve the kettle control by adding some compensation circuits as shown in Fig. 5.38.

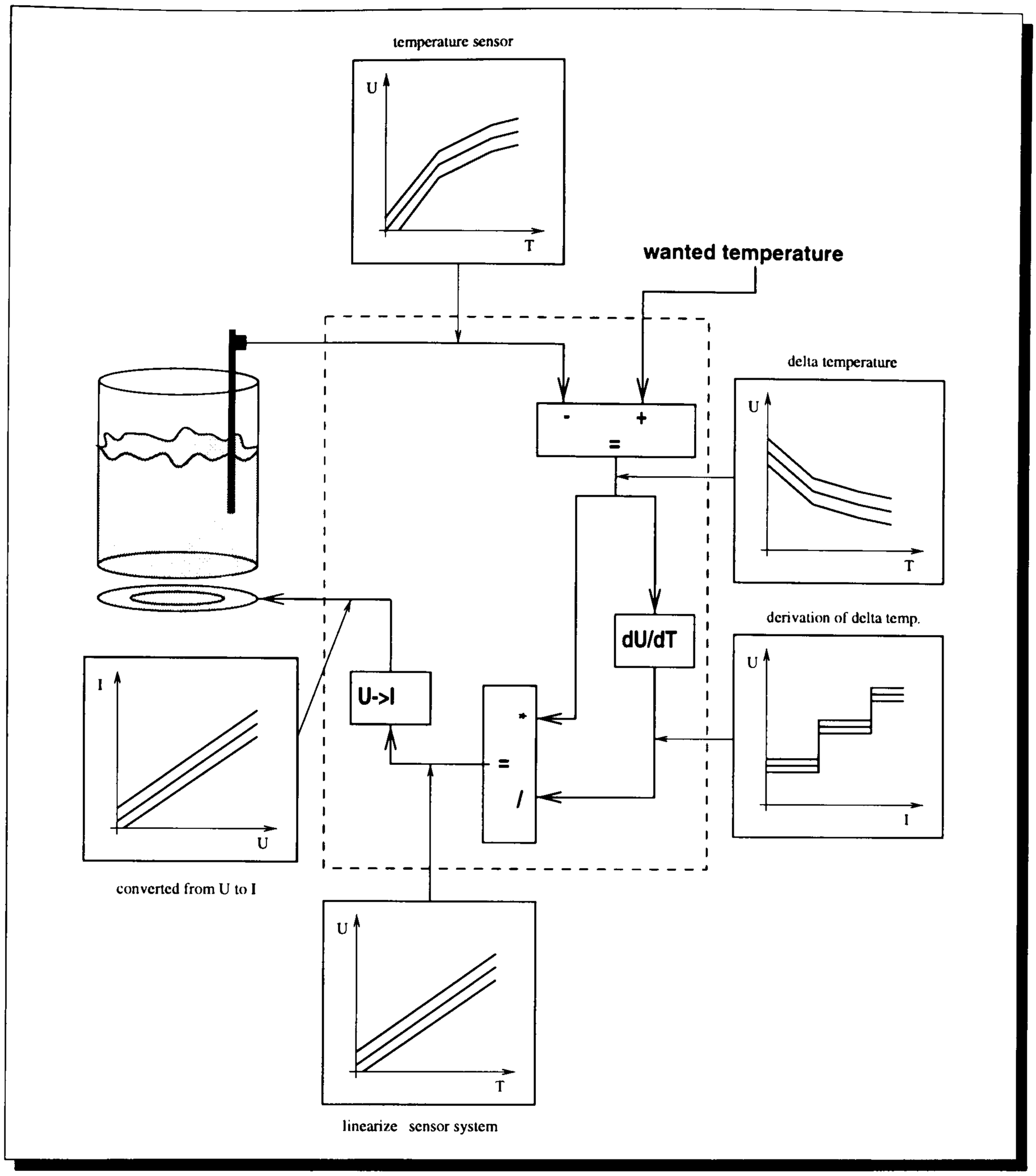


Figure 5.38: Intelligent Kettle Control

In Fig. 5.38 the nonlinearity of the `temperature sensor` is displayed. The signal measured at the `temperature sensor` is guessed and represented by a Fuzzy Curve taking into account that the behaviour of the sensors is varying mainly caused by the manufacturers. Often it can be said, that the lower the cost of a sensor device, the larger the deviations of the sensor specifications. One possibility of representing

the `temperature sensor` and the `heating system` is define at the α-cut level 1.0 the curve that represents most likely the systems and at α-cut level > 0.0 the maximum/minimum data specifications of the systems.

The sensor system of the kettle is linearized by dividing the derivated difference of the `wanted temerature` and the measured temperature. An overview of the measured fuzzy curves is shown in Fig. 5.38. Fig. 5.39 shows the `temperature sensor` curve in more detail.

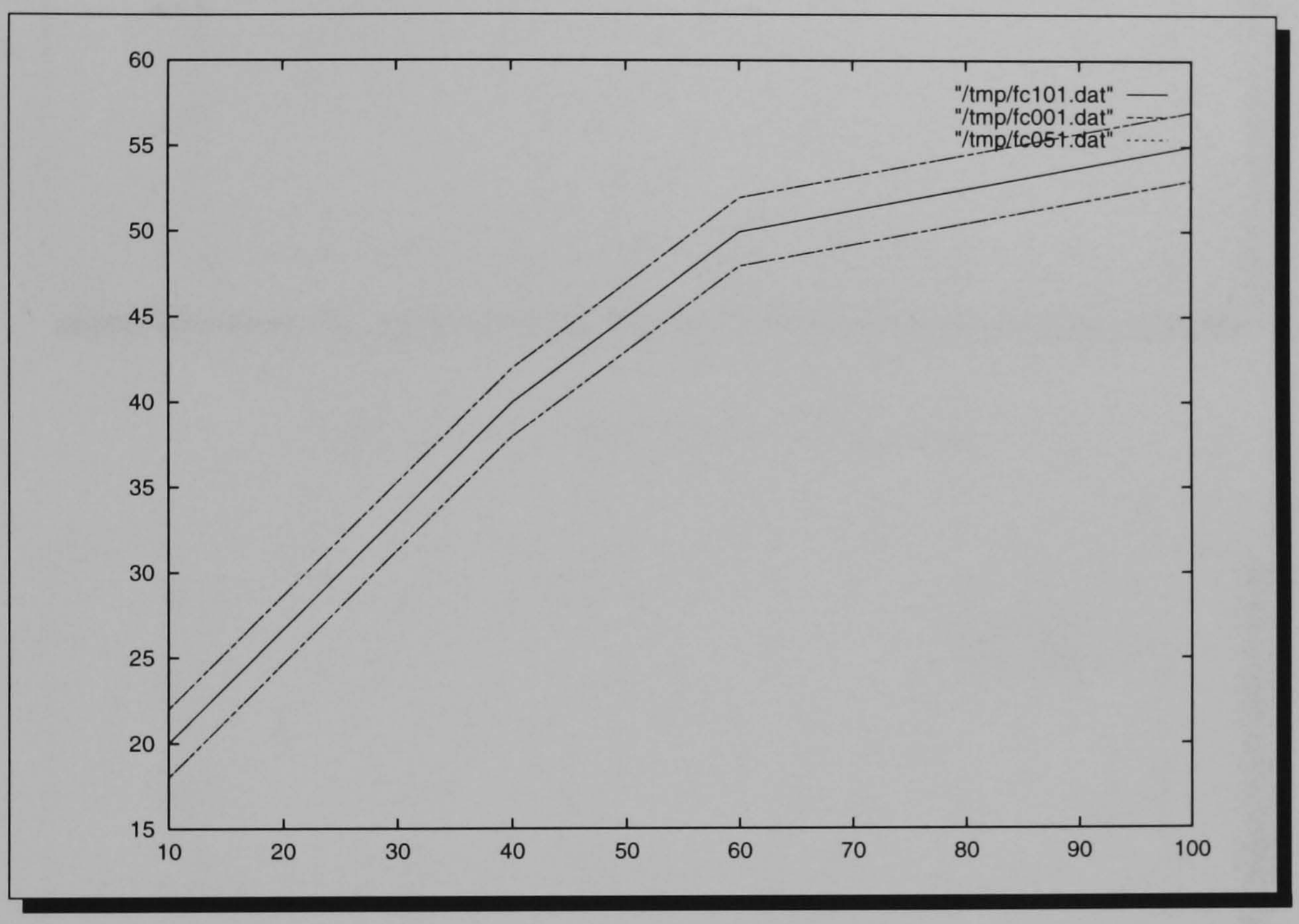


Figure 5.39: Temperature Sensor Curve

Subtracting the `wanted temperature` given by a fuzzy number $\tilde{N} = (50, 2, 2)$ and derivate the resulting fuzzy curve the fuzzy curve shown in Fig. 5.40 is generated. To get a more or less linear function 5.41 the subtraction is divided by the derivation Fig. 5.40.

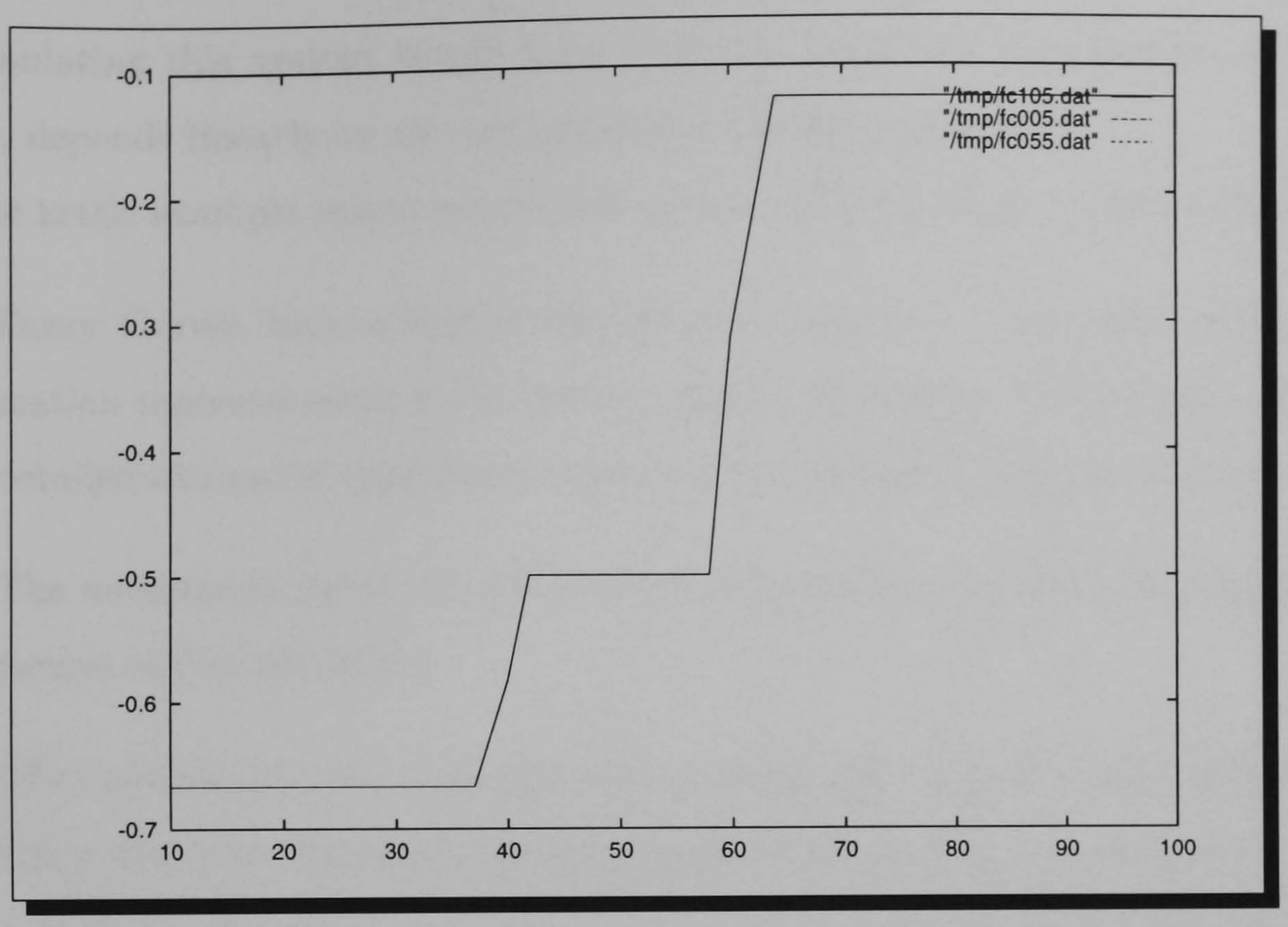


Figure 5.40: Derivation of Sensor Curve

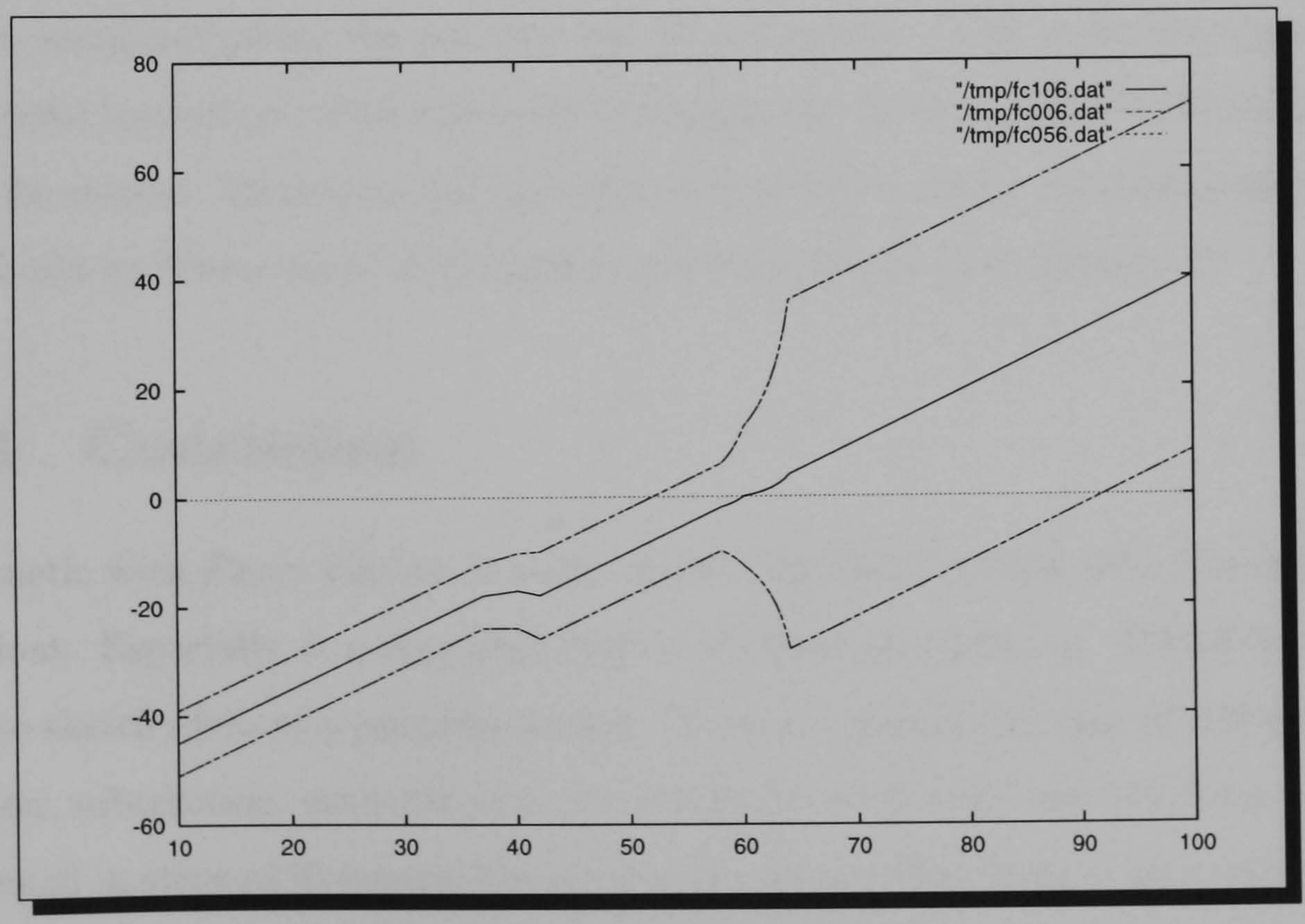


Figure 5.41: Linearized Kettle Control

Simulating this system would show that the measured temperature, with the sensor, depends linearly on the temperature of heating system.

The kettle example states several advantages of using Fuzzy Relation Memories:

1. Fuzzy Curves have a highly interpretive character. Generally visual information representation is for humans more informative (See Chapter 11 for a detailed discussion especially related to the design of analogue circuits.).

2. The uncertainty involved of the sensor characteristic at different temperature ranges can be modelled.

3. After simulating the nonlinear system using the mathematical operators for Fuzzy Relation Memories the robustness of the system is demonstrated. Fig. 5.41 shows three curves. The curve in the center represents the output of a kettle control using a typical kettle control model. The upper and lower curves show the possible outcome of the kettle control system when the model is simulated using the extreme model parameters. The wider the upper and lower boundary curves are apart the bigger the uncertainty and the less robust the system. Mathematical operators are mostly used for combining models to build up hierarchical structures of a system model (see Chapter 7).

5.14 Conclusion

Arithmetic with Fuzzy Curves is more natural for humans then with time series or equations. Especially at a very high-level of abstraction when, e.g. design engineers, want to sketch ideas of a possible system. There is a computational advantage when addition, subtraction, multiplication, or division is used, since the calculation can be performed in steps of Temporal Fuzzifying Functions. The Fuzzy Curves are treated as piecewise linear functions for the different α-levels. By performing derivation or integration this advantage is lost. The Fuzzy Curves have to be made discrete and

built up again. By using a rough approximation, error is introduced. There could be a lot more defined, like sin, cos, etc., which is out of the scope of this work.

Chapter 6

Simulation of Imprecise Ordinary Differential Equations

Differential equation problems occur in many different technical disciplines. Many mathematical models derived in the study of physical systems involve instantaneous rates of change that are given mathematically by the derivatives of one variable with respect to another. None the less engineers are used to model their dynamic nonlinear physical systems by differential equations.

This chapter starts with an introduction of solving ordinary differential equations and demonstrates the problems given imprecise initial values or imprecise differential equation parameters. It discusses different approaches for solving imprecise differential equations and describes the new approach: **Interactive Evolutionary Algorithm Approach**. Finally some examples are given and a conclusion is drawn.

6.1 Initial Value Problems

Differential equations relate the derivatives of variables and functions of these variables. For example, consider the circuit diagram given in Fig.6.1.

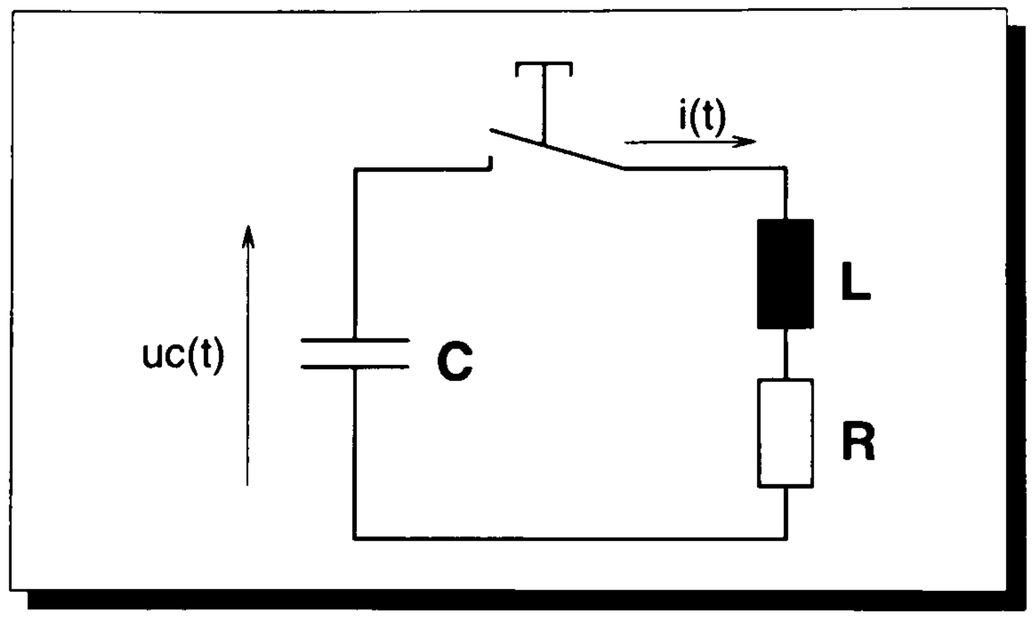


Figure 6.1: RLC Series Circuit

The diagram represents a simple electrical circuit involving a resistance R in series with an inductance L and a capacitor C. Initially at time $t = 0$ the switch is open and the current is consequently zero. A physical analysis of the circuit leads to the following differential equation giving the current at any time t.

$$L * C * \frac{du_c^2(t)}{dt^2} + R * C * \frac{du_c(t)}{dt} + u_c(t) = 0 \tag{6.1}$$

with the initial conditions:

$$\begin{aligned} u_c(0) &= voltage\ of\ charged\ capacitor, \\ \frac{du_c(0)}{dt} &= 0\frac{V}{s} \end{aligned} \tag{6.2}$$

This second order differential equation belongs to the category of problems called initial value problems. Initial values are given for the dependent variable and its derivatives which allow a specific solution to be found.

6.2 Taylor's Series

Methods for solving the initial value problem are based on the Taylor's series ([Lindfield and Penny, 1989]). The Taylor series is an approximation of the exact solution of the differential equation. The Taylor series expansion of the function $y(0)$ about the point x_0 is given by

$$y(x) = y_0 + (x - x_0) * y_0' + \frac{(x - x_0)^2 * y_0''}{2!} + ... + \frac{(x - x_0)^n * y_0^{(n)}}{n!} + ... \tag{6.3}$$

where y_0' represents the first derivative, y_0'' the second derivative and $y_0^{(n)}$ the nth derivative of y with respect to x evaluated at $x = x_0$.

6.3 Runge-Kutta Method

An important group of methods for obtaining the numerical solution of a differential equation are the Runge-Kutta methods [Bronstein and Semendjajew, 1991]. In contrast to other methods, e.g. Euler Method, Modified Euler, etc., the Runge-Kutta Method is a single step method. With the Runge-Kutta Method any degree of accuracy may be obtained at each step but the better the accuracy the greater the number of function evaluations required. In this thesis the classical 4-order Runge-Kutta Method has been chosen which has the following form

$$y_{n+1} = y_n + \frac{k_1 + 2 * k_2 + 2 * k_3 + k_4}{6} \tag{6.4}$$

with

$$\begin{aligned} k_1 &= h * f(x_n, y_n) \\ k_2 &= h * f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 &= h * f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 &= h * f(x_n + h, y_n + k_3) \end{aligned} \tag{6.5}$$

where h is the step size. The Runge-Kutta method solving n-th order differential equations

$$dy^n + a_{n-a} * dy^{n-1} + ... + a_2 * dy^2 + a_1 * dy + a_0 * y = f(x) \tag{6.6}$$

with the initial condition $dy^n(x_0)$, ..., $dy^2(x_0)$, $dy(x_0)$, $y(x_0)$. After substitution as follows

$$\begin{aligned} z_1 &= y \\ z_2 &= dz_1 = dy \\ z_3 &= dz_2 = dy^2 \\ &\cdot \\ &\cdot \\ &\cdot \\ z_{n-1} &= dz_{n-2} = ... = dz_2^{n-1} = dz_1^n = dy^n \end{aligned} \tag{6.7}$$

the differential equation can be written in matrix form.

$$\underline{z}'(x) = \underline{A} * \underline{z}(x) + \underline{B} * f(x) \tag{6.8}$$

with

$$\underline{z}'(x) = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ . \\ . \\ z_{n-2} \\ z_{n-1}n \end{bmatrix}' \quad \underline{A} = \begin{bmatrix} 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 1 & ... & 0 \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ 0 & 0 & 0 & ... & 1 \\ a_0 & a_1 & a_2 & ... & a_{n-1} \end{bmatrix} \quad \underline{z}(x) = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ . \\ . \\ z_{n-2} \\ z_{n-1} \end{bmatrix} \quad \underline{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ 0 \\ 1 \end{bmatrix}$$

Given the initial vector $\underline{z}_0$ the Runge-Kutta Method of order four is defined as

$$\begin{aligned}
\underline{K}_1 &= h * [\underline{A} * \underline{z}(k*h) + \underline{B} * f(k*h)] \\
\underline{K}_2 &= h * [\underline{A} * (\underline{z}(k*h) + \frac{1}{2} * \underline{K}_1) + \underline{B} * f((k+\frac{1}{2})*h)] \\
\underline{K}_3 &= h * [\underline{A} * (\underline{z}(k*h) + \frac{1}{2} * \underline{K}_2) + \underline{B} * f((k+\frac{1}{2})*h)] \\
\underline{K}_4 &= h * [\underline{A} * (\underline{z}(k*h) + \underline{K}_3) + \underline{B} * f((k+1)*h)] \\
\underline{z}_{k+1} &= \underline{z}_k + \frac{1}{6} * [\underline{K}_1 + 2 * \underline{K}_2 + 2 * \underline{K}_3 + \underline{K}_4]
\end{aligned} \tag{6.9}$$

A disadvantage is that the Runge-Kutta Method requires a considerable number of function evaluations at each step. But the Runge-Kutta method has less function evaluations than multi step methods and has a greater accuracy than e.g. the Euler Method.

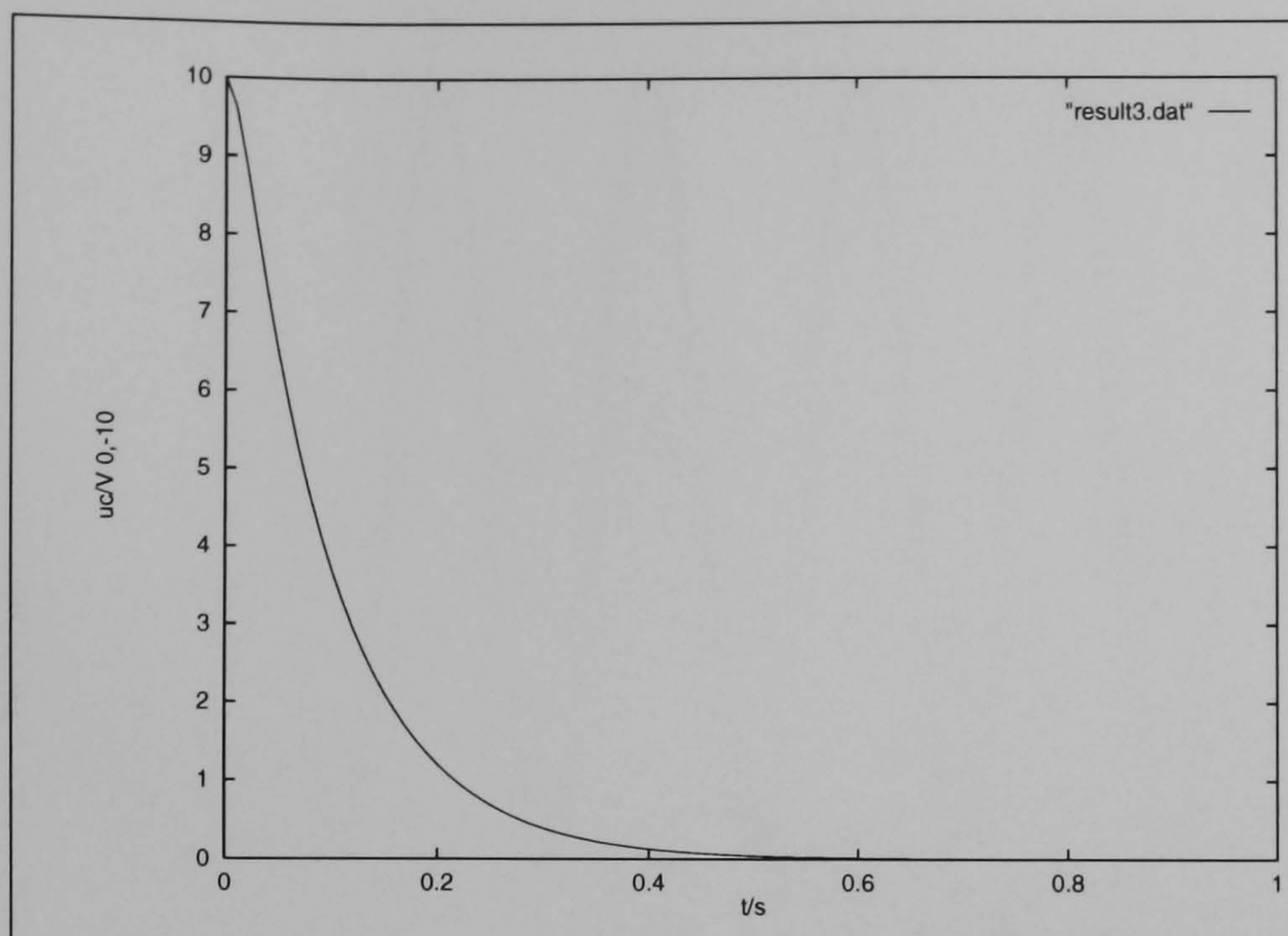
6.3.1 Example: 2nd Order Differential System

The example shown in Fig. 6.1 is described by

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -\frac{1}{L*C} & -\frac{R*C}{L*C} \end{bmatrix} * \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} * 0 = \begin{bmatrix} 0 & 1 \\ -\frac{1}{L*C} & -\frac{R*C}{L*C} \end{bmatrix} * \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

Simulating the system with Runge-Kutta can result into three different behaviours given different parameter values: the aperiodic (Fig.6.2), the periodic with attenuation (Fig.6.3), and the periodic without attenuation (Fig.6.4) system behaviour.

Aperiodic system behaviour

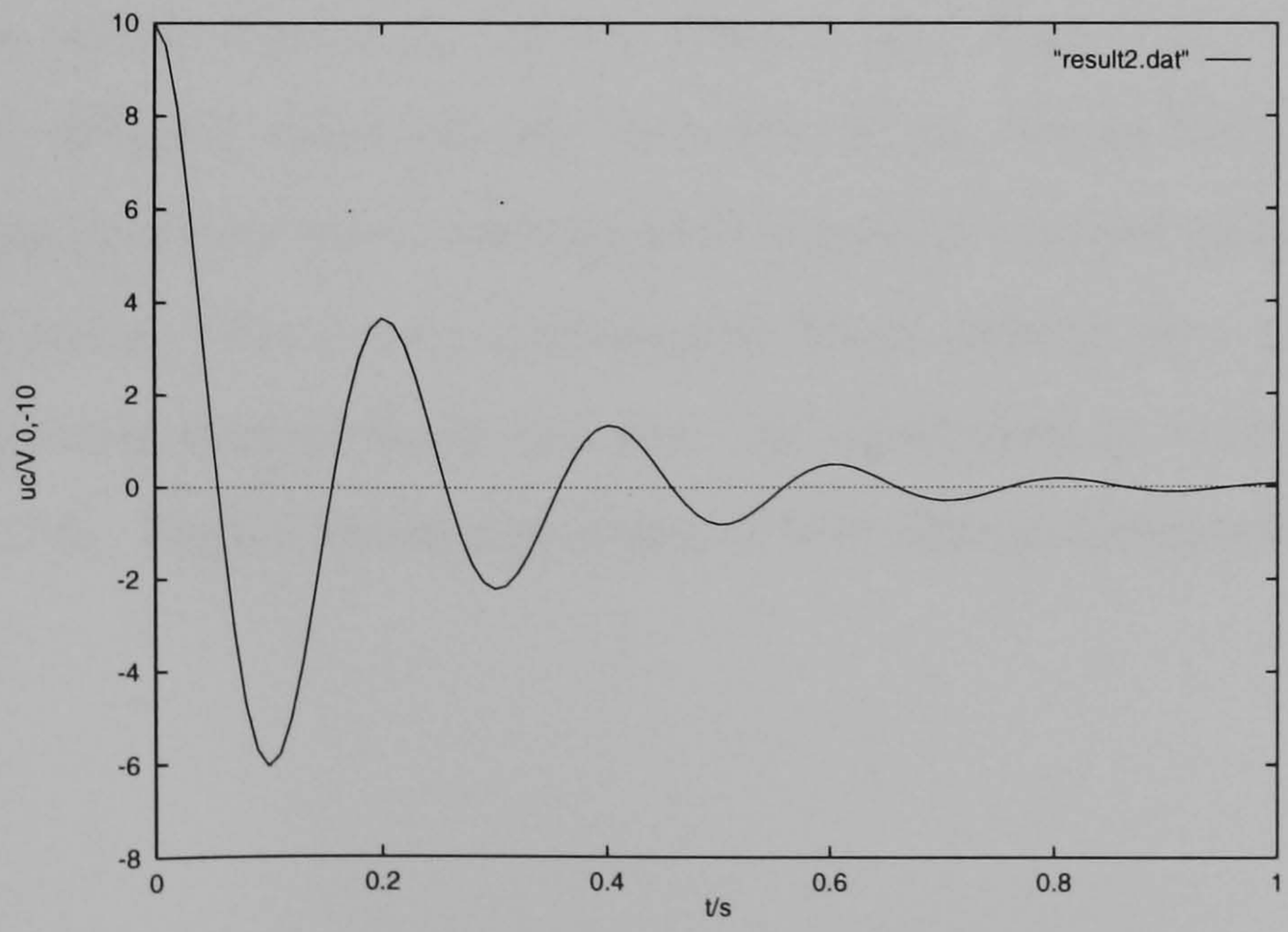


Given:

$$R = 100\Omega$$
$$L = 1H$$
$$C = 1mF$$
$$uc(0) = 10V$$
$$uc(0)' = 0$$
$$h = 0.01$$
$$x_0 = 0$$
$$x_n = 1$$

Figure 6.2: Aperiodic System Behaviour

Periodic system behaviour with attenuation



Given:

$$R = 10\Omega$$
$$L = 1H$$
$$C = 1mF$$
$$uc(0) = 10V$$
$$uc(0)' = 0$$
$$h = 0.01$$
$$x_0 = 0$$
$$x_n = 1$$

Figure 6.3: Periodic System Behaviour With Attenuation

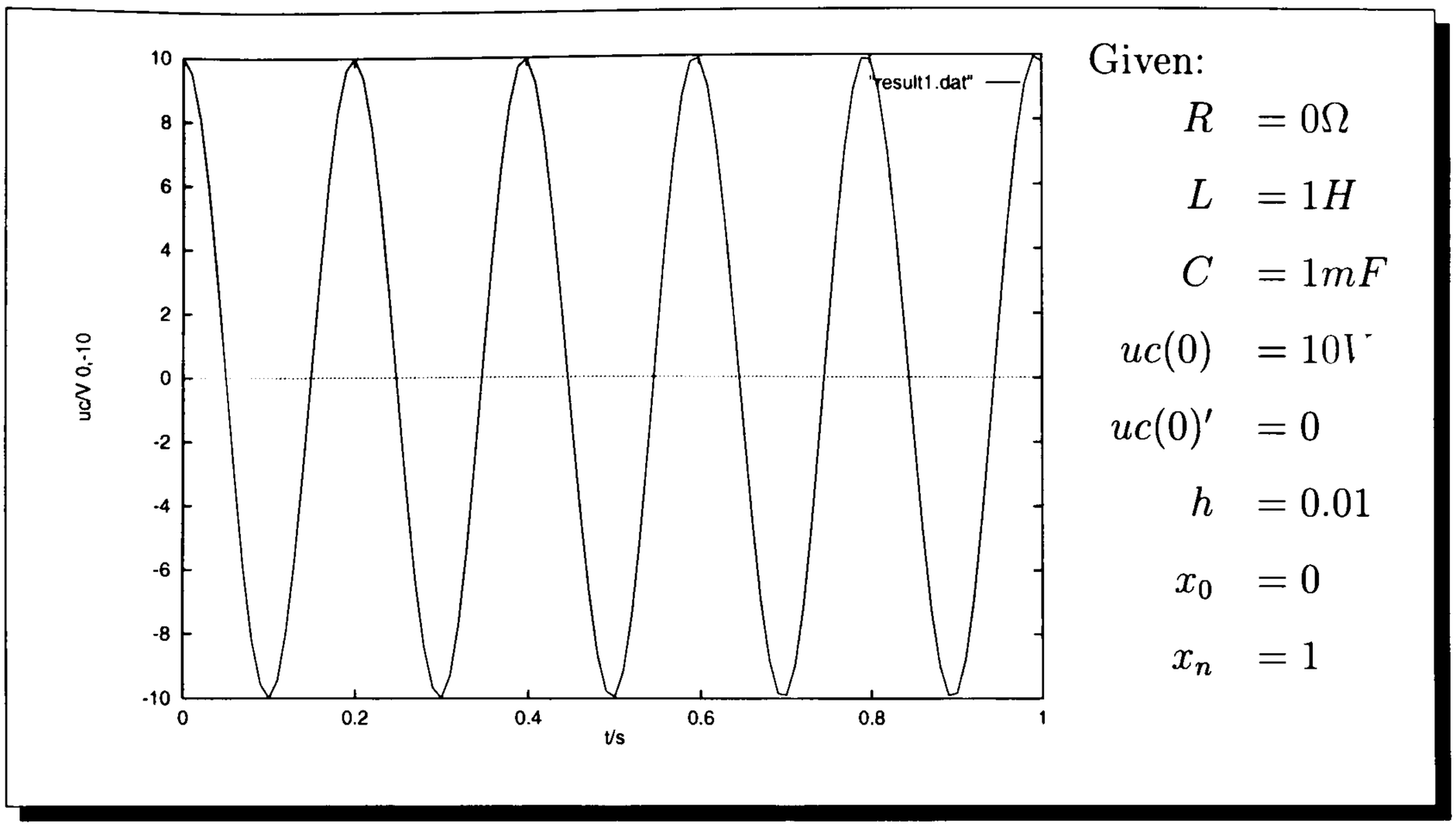


Figure 6.4: Periodic System Behaviour Without Attenuation

6.4 Problems Working With Imprecise Values

The results shown in Fig.6.2, Fig.6.3, and Fig.6.4 of the differential equation are very different although only the value of the resistor has been changed. Exactly this is the problem when working with imprecise defined values. Suppose the resistor of the circuit (Fig.6.1) is a triangular fuzzy number $\tilde{R} = (80, 80, 20)$ which is a very imprecise defined fuzzy number. At α-cut level ≥ 0 the interval of uncertainty is $[0, 100]$. Fig.6.5 shows the result of the differential equation in 10Ω-steps.

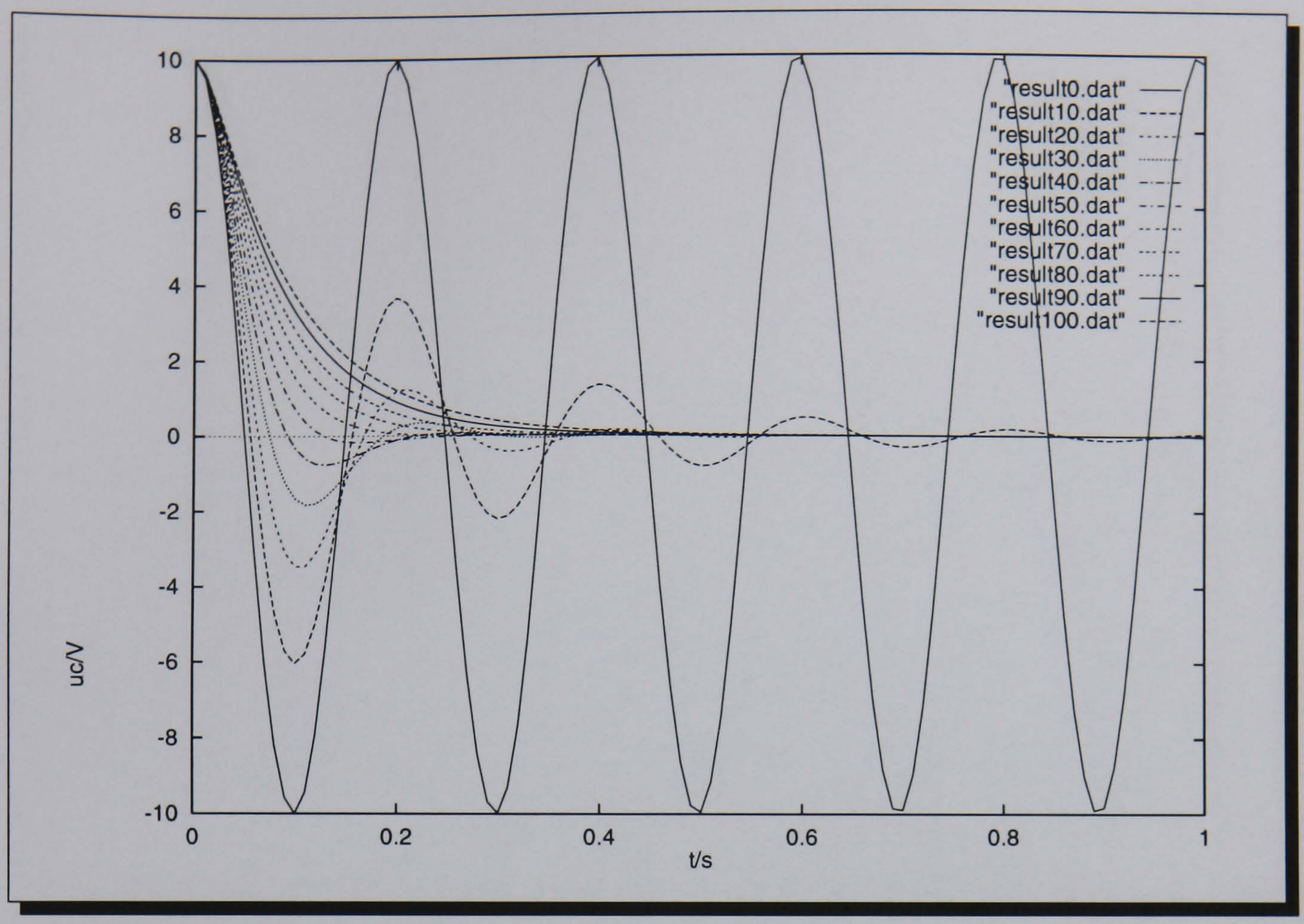


Figure 6.5: Result of: $0.001 * \frac{du_c^2(t)}{dt^2} + R * 0.001 * \frac{du_c(t)}{dt} + u_c(t) = 0$ with $R = 0, 10, 20, ..., 100$

Fig. 6.5 shows a wide range of possible solutions of the differential equation. In this dissertation a solution of a differential equation is emphasized that represents the minimum and the maximum of all possible solutions. What is needed is a minimal and a maximal curve in between all the possible solutions of the differential equations belong. For example as shown in Fig.6.6 for the particular differential equation.

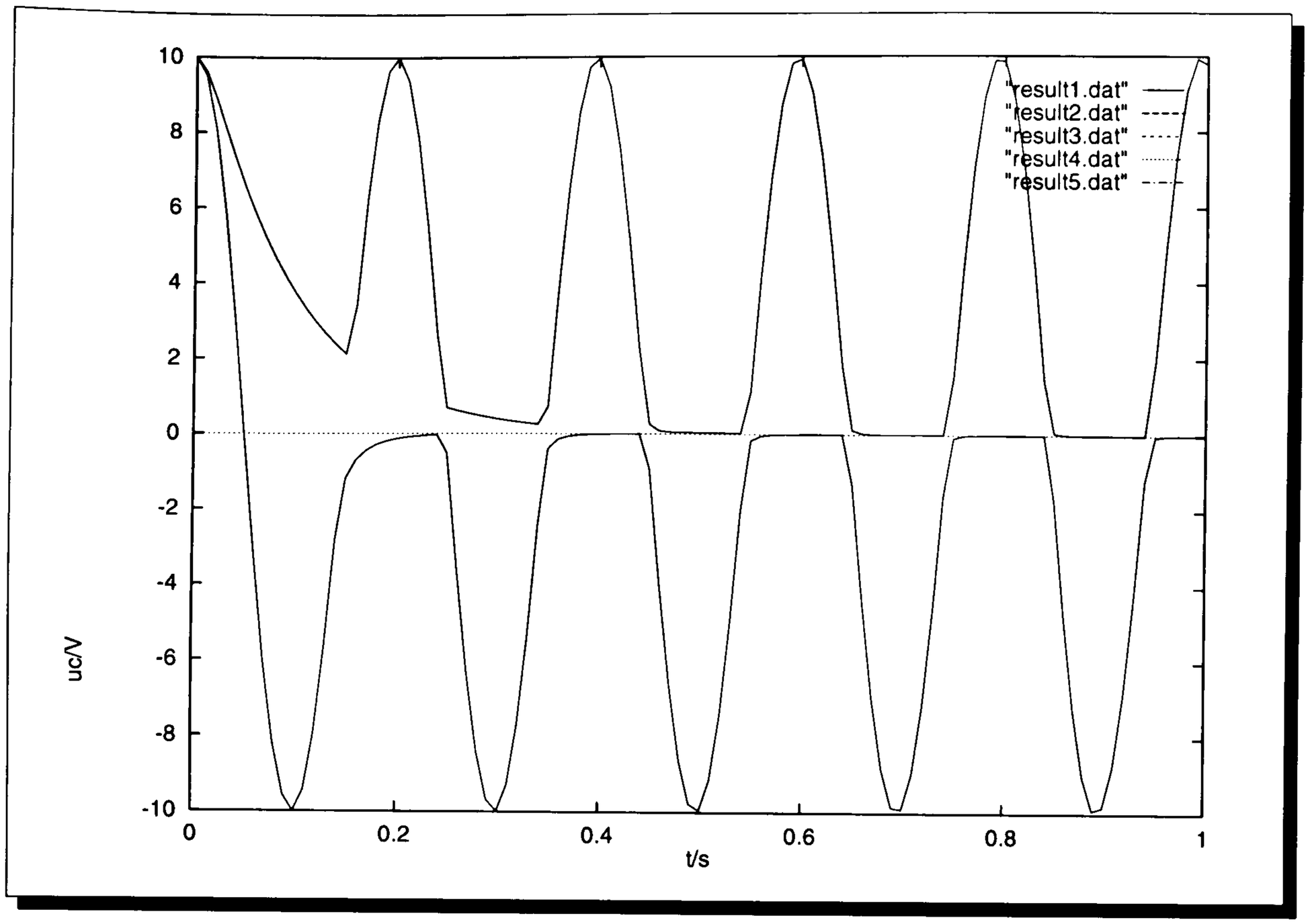


Figure 6.6: Min/Max Result of: $0.001 * \frac{du_c^2(t)}{dt^2} + R * 0.001 * \frac{du_c(t)}{dt} + u_c(t) = 0$ with $R = 0, 10, 20, ..., 100$

6.5 Existing Approaches for Solving Imprecise Differential Equations

In the following sections (6.5.1, 6.5.2, 6.5.3) there are three approaches discussed that are able to solve ordinary differential equations which have imprecise defined initial conditions or imprecise equation parameters. They are:

- Interval Arithmetic Approach
- Non Interacting Approach
- Interacting Approach

Bonarini and Bontempi [Bonarini and Bontempi, 1994] developed two approaches, the "non interacting" and the "interacting" approach.

6.5.1 Interval Arithmetic Approach

The coefficients of the Taylor's series for a numeric differential system are numeric coefficients. Using fuzzy or qualitative values represented by α-cuts for the initial condition makes it necessary to extend the Taylor's series to intervals at different level of presumptions. The Taylor's series becomes an algebraic sum of expressions each one of which takes an interval value.

$$\begin{aligned}[ymin, ymax](x) =& [ymin_0, ymax_0] + (x - [xmin_0, xmax_0]) * [ymin_0, ymax_0]' + \\ & \frac{(x - [xmin_0, xmax_0])^2 * [ymin_0, ymax_0]''}{2!} + \ldots + \\ & \frac{(x - [xmin_0, xmax_0])^n * [ymin_0, ymax_0]^{(n)}}{n!} + \ldots \end{aligned} \tag{6.10}$$

Using interval arithmetic the width of the result is always equal to the sum of the widths of the two expressions. As a result, the width of $[ymin, ymax](x)$ becomes wider with x increasing. In the case of a dynamic system, where the independent variable x is the time, the system output would be affected by an always increasing width, independent of stability of other properties of the system. Most likely the simulation result will reach regions of the phase space where no numerical solution of the differential system pass through. Therefore the system simulation based on interval arithmetics would produce output in contradiction with the physical reality described by the model.

6.5.2 Non Interacting Approach

The *Non Interacting Approach* of Bonarini and Bontempi [Bonarini and Bontempi, 1994] considers the variables as always non interact-

ing. All interval values N generate at $t = 0$ an N-cube. Every point of the hyper-cube is the initial condition of a solution of the differential equation system. After an ordinary numeric integration till the instant $t + \Delta t$ the non interacting N-cube at the instant $t + \Delta t$ is reconstructed. This is done for each α-level. The important drawback is the insertion of spurious trajectories which do not belong to the system behaviour. Fig. 6.7 shows the result of the differential equation

$$L * C * \frac{du_c^2(t)}{dt^2} + R * C * \frac{du_c(t)}{dt} + u_c(t) = 0 \tag{6.11}$$

with just one fuzzy number $\tilde{R} = (80, 80, 20)$

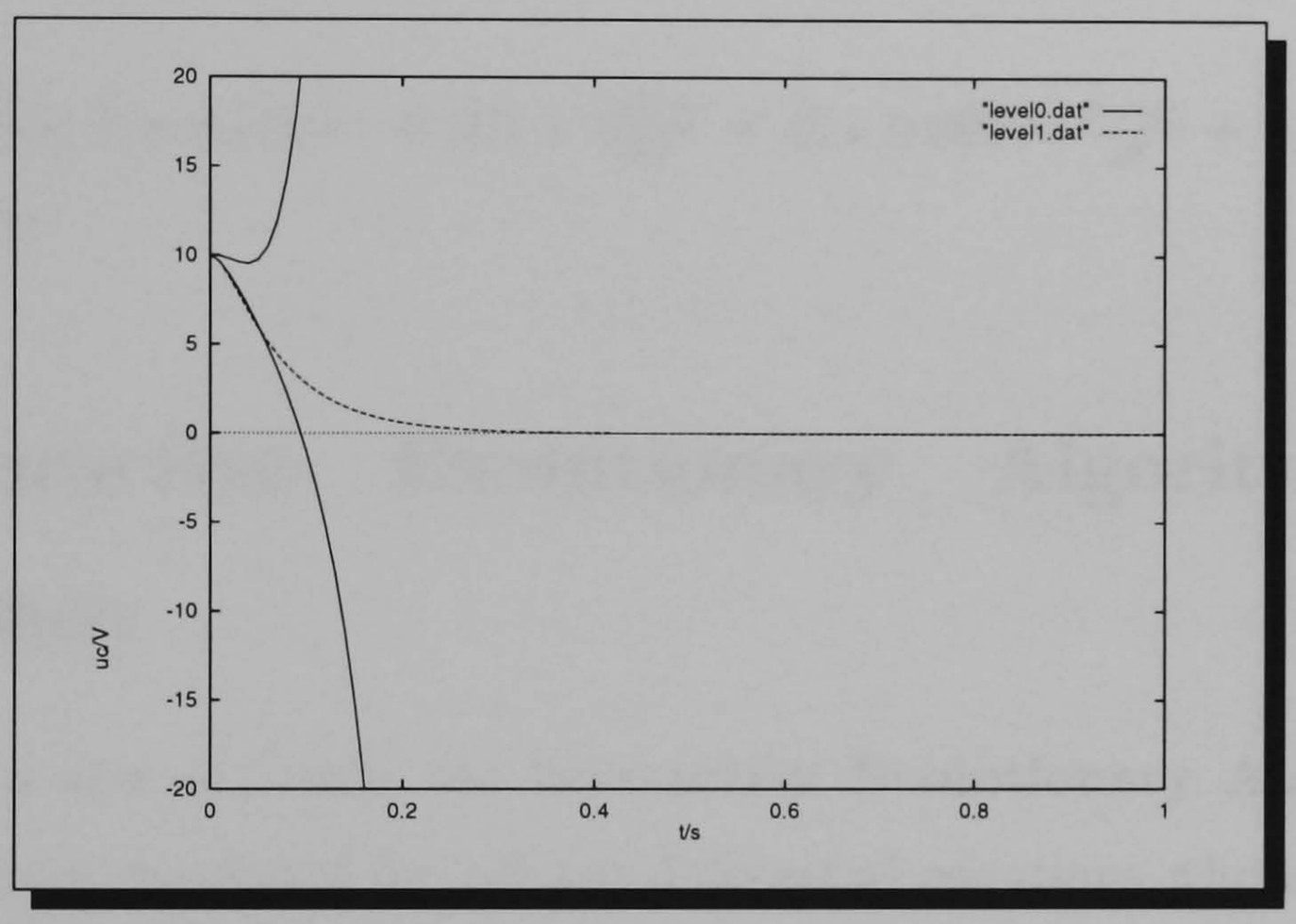


Figure 6.7: Non Interactive: $0.001 * \frac{du_c^2(t)}{dt^2} + \tilde{R} * 0.001 * \frac{du_c(t)}{dt} + u_c(t) = 0$ with $\tilde{R} = (80, 80, 20)$

6.5.3 Interacting Approach

The *Interacting Approach* of Bonarini and Bontempi [Bonarini and Bontempi, 1994] states that the uncertainty region (N-cube) is not approximated at each integration step. The variables are interacting for the whole simulation process. A drawback is the neglection of possible trajectories. Using the same differential equation as in the previous section 6.5.2 the result looks like Fig. 6.8:

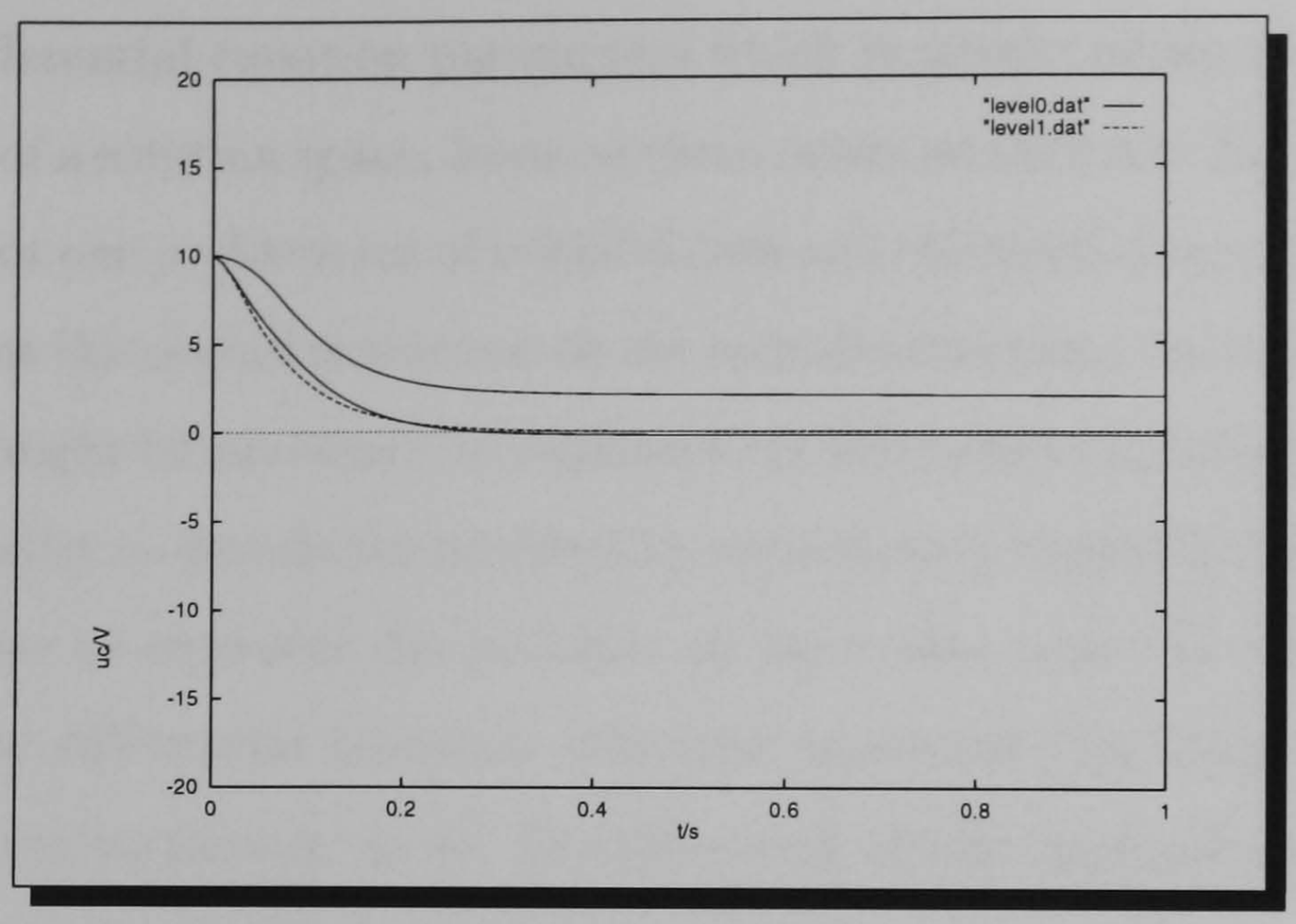


Figure 6.8: Non Interactive: $0.001 * \frac{du_c^2(t)}{dt^2} + \tilde{R} * 0.001 * \frac{du_c(t)}{dt} + u_c(t) = 0$ with $\tilde{R} = (80, 80, 20)$

6.6 Interactive Evolutionary Algorithm Approach

In this thesis a new approach, the **Interactive Evolutionary Algorithm Approach**, has been developed for solving differential equations whose initial values and differential equation parameters are not known exactly. The initial values and the parameters of the differential equation are represented by fuzzy values. For every α-cut level the Interactive Evolutionary Algorithm Approach is applied twice; searching first for the minimum curve and second for the maximum curve. In between these two curves lie all the possible solutions of a differential equation. Finding the minimum or maximum curve which includes possible solutions at each α-cut level is not a trivial task. Evolutionary algorithms have shown that they have been very successful in finding optima for complex tasks.

The classic evolutionary algorithm approach is suitable for finding an optimal solution for the initial value problem but is not suitable for finding the set of initial

values and differential equation parameters which represent either the maximum or the minimum of a solution space, because there exists no such set. Fig. 6.6 illustrates that there is not one problem set of initial values and differential equation parameters which represent the overall maximum or the overall minimum. An infinite number of problem sets might be necessary to represent all solutions of a differential equation.

One possibility to decode the problem by evolutionary algorithms is by expanding the chromosome to represent the problem set more then once, theoretically infinite size. Then the differential equation solutions represented by these sets build the overall minimum/maximum curve. The drawback of this approach is the increasing number of calculations necessary.

The **Interactive Evolutionary Algorithm Approach** codes the parameters of the differential equation just once by the chromosome and searches for the minimum/maximum. During this search the chromosomes get evaluated several times using the Runge-Kutta Method (see Section 6.3). The solution curve determined by the Runge-Kutta Method is represented by x-y value pairs. Each time, the calculated values are compared with the overall minimum/maximum value pairs of a database. The first time the empty database is filled with the first calculated x-y values. Later the calculated values are saved when they are smaller/bigger than the actual minimum/maximum value stored in the database. When no single problem set of initial values and differential equation parameters exists, the *Interactive Evolutionary Algorithm* jumps from one possible problem set to the other recording the overall minimum/maximum value in the database.

6.6.1 Classic Evolutionary Algorithm

> Natural evolution has been powerful enough to bring about biological phenomena as complex as mammalian organisms and human consciousness [Duprè, 1987].

Engineers have been inspired to learn from nature and to apply her recipes in tech-

nology. The classic evolutionary algorithm is a program for an artificial evolution process which simulates some of the principles of natural evolution. The evolutionary algorithm used in this thesis has been first described by I. Rechenberg [Rechenberg, 1973], [Rechenberg, 1994], and H.-P. Schwefel [Schwefel, 1995]. The algorithm simulates the most import mechanisms of natural evolution which are:

- *Natural selection* favours reproduction of chromosomes with successful structures.
- *Mutation* generates new variants of chromosomes.
- *Reproduction* causes a fast spread of successful mutations in the population. Sexual reproduction might recombine several successful mutations.

The basic evolutionary algorithm can be described by the following steps:

1. Initializing of a random generated population (equal to set of individuals or set of chromosomes).
2. Evaluate each individual of the population.
3. Test if stop criteria is reached (e.g. number of generation). If not do the following until stop criteria is reached.
   (a) Select individuals from the parents for reproduction and recombine them generating the children.
   (b) Mutate the children.
   (c) Evaluate the children.
   (d) Select the fittest from the children or select the fittest from children and parents and build a new population.

Evolutionary strategies use natural evolution as a guiding principle for optimization problems. Variations of possible solutions are generated and the best solution is

selected iteratively until the optimum is found or the stop criteria is reached. The search for optimum by evolutionary strategies is searching through the solution space applying the mechanisms of natural evolution [Banzhaf et al., 1998].

6.6.2 Interactive Evolutionary Algorithm

Fig. 6.9 visualizes the principle of the approach taken in this work using the graphical notation developed by Rechenberg. The algorithm is called interactive because the objective function assumes that the fuzzy values are interactive during the evaluation of the chromosomes.

From an initial population; parent population $1..n$; m children are generated by making copies from parents selected by random and mutating them. From m evaluated children the best of the children and the parents are selected to build a new population. The process of generating mutated children from parents and select, the best from both is repeated several generations.

Both the minimum and the maximum should be optimized. Therefore the evolutionary process must run twice. Contrary to the classic evolutionary algorithm is the addition of the database which records the absolute minimum/maximum values of the evaluation process (see Section 6.6.5 for more detail).

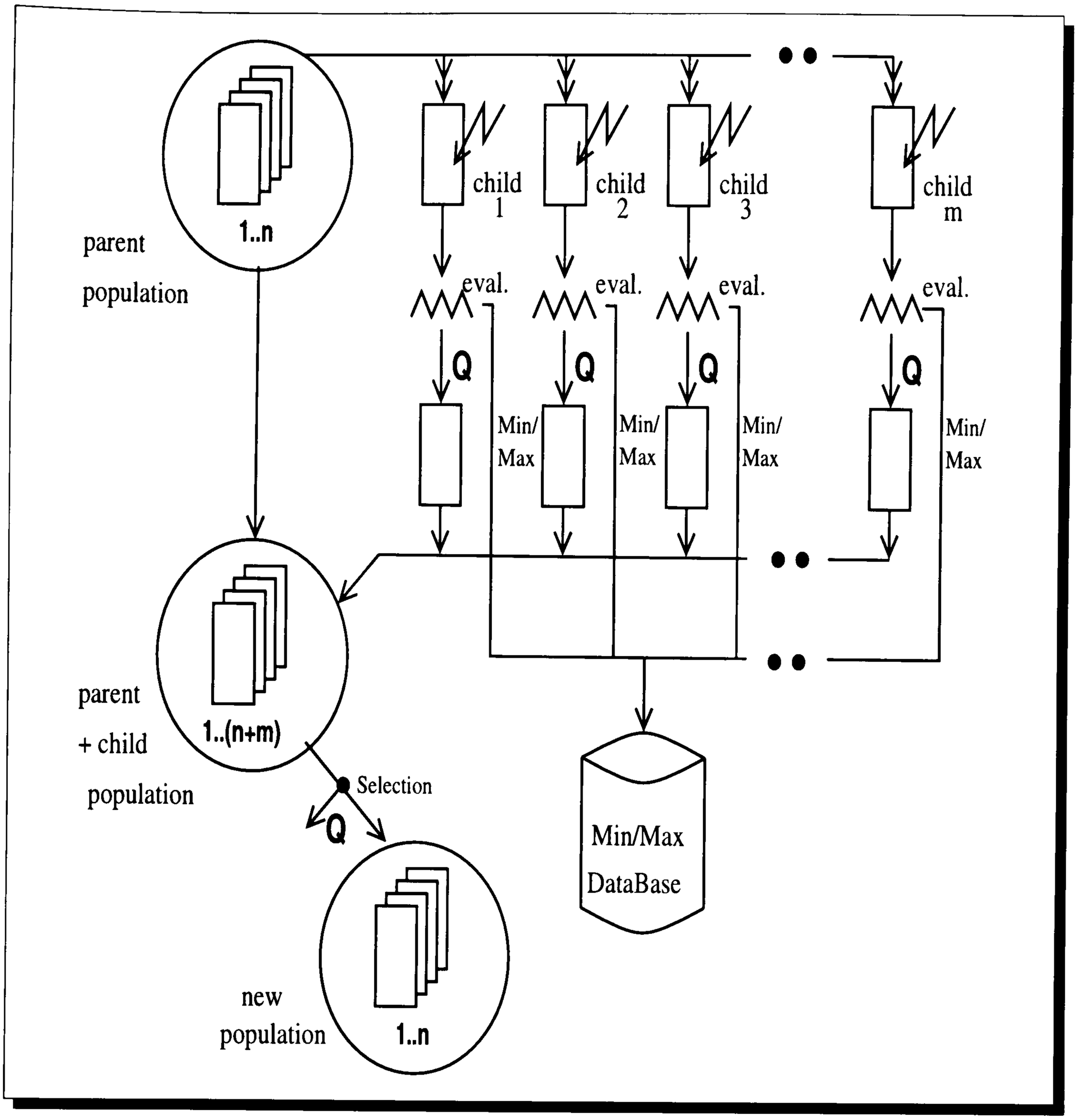


Figure 6.9: Interactive Evolutionary Algorithm

6.6.3 Representation of Population

A population is represented by several individuals in the chromosomes. The initial values and the parameters of the ordinary differential equation are represented by a chromosome. The chromosome is a n vector of real values. It is built up as follows:

value 1	value 2	value 3	...	value n

For example, the following first order differential equation

$$a_1 * f'(t) + a_0 * f(t) = b$$

is represented as:

a_0	a_1	b	$f(0)$

6.6.4 Mutation of Chromosomes

The mutation is the most essential part of the evolutionary algorithm.

Definition 6.58 (*Mutation*): Random change of chromosomes.

If a chromosome is mutated one value of the vector $1...n$ of the chromosome is changed. To give the overall algorithm a direction in searching for the optimum, the maximum change each generation of a value must be limited.

Definition 6.59 (*Maximum Change Parameter*): Each generation a chromosome's value can only be changed by a value defined by its *maximum change parameter.*

Mutation generates a random value from the interval

$$[-\text{maximum change parameter}, \text{maximum change parameter}].$$

Without *maximum change parameter*, there is a big risk that the process is not settling at all; not finding any optimum. Because each value of the chromosome has a different meaning and each value has its own *maximum change parameter.*

A *mutation strength* has been defined to set the *maximum change parameter* during the optimum process.

Definition 6.60 (*Mutation Strength*): The *mutation strength* defines the percentage of the *maximum change parameter* used in each generation.

If the *mutation strength* is 80% the interval for picking the random value to change the chromosome value is

$$[-0.8 * \text{maximum change parameter}, 0.8 * \text{maximum change parameter}].$$

This enables the evolutionary algorithm to adapt mutation. At the beginning of the optimum process the *mutation strength* is high to avoid getting stuck at local optimum.

To automate the adaption of the *mutation strength* Rechenberg's $\frac{1}{5}$-Rule-of-Success as stated in [Schoeneburg et al., 1994] has been applied.

Definition 6.61 ($\frac{1}{5}$-*Rule-of-Success*): $\frac{1}{5}$ of the mutated children must be better then the parents otherwise the *mutation strength* must be adapted. If more than $\frac{1}{5}$ of the mutated children are better than their parents, the mutation strength is increased. If less than $\frac{1}{5}$ of the mutated children are better than their parents, the mutation strength is decreased.

None the less a *mutation rate* is needed to state the overall number of mutations that each generation could occur.

Definition 6.62 (*Mutation Rate*): Defines how often a chromosome gets mutated at each generation.

6.6.5 Objective function

The chromosomes are evaluated using the classic Runge-Kutta method defined in Section 6.3. The result of the Runge-Kutta method is a function represented by pairs of x-y-values.

Definition 6.63 (*Objective Function for Interactive Evolutionary Algorithm Approach*): Given pairs of

$$x_i \text{ and } y_i \text{ with } i = 0, 1, 2, ..., n \tag{6.12}$$

n is the maximum number of steps used to approximate the result of the differential equation by the Runge-Kutta method. From this follows the objective function:

$$Q = \sum_{i=0}^{n} y_i \text{ with } i = 0, 1, 2, ..., n \tag{6.13}$$

The sum of all y-values is the objective function that determines the evaluation value for each chromosome.

The *Interactive Evolutionary Algorithm* is looking for the minimum or maximum solution for the differential equation respectively. At the end of the evolutionary process the optimal chromosome should represent the minimum/maximum function determined by the Runge-Kutta Method. But it is impossible to tell whether another chromosome could have generated a better optimum for a particular x-y-value pair during the optimizing process or not. The overall minimum/maximum x-y-values are retrieved by protocolling them during the evolutionary algorithm process shown in Fig. 6.9.

An advantage is that it is sufficient to represent one problem set — initial values and parameters of the differential equation — in a chromosome. If there is more than one problem set necessary to represent the minimum/maximum curve the evolutionary process jumps between these solutions back and forth protocolling the minimum/maximum x-y-pairs.

A disadvantage is that each generation the Runge-Kutta method is used to evaluate the individuals which is computationally expensive.

6.6.6 Detailed Example: 1st-Order Differential Equation

The following 1st-order differential equation has to be solved:

$$a1 * y' + a0 * y = 0$$

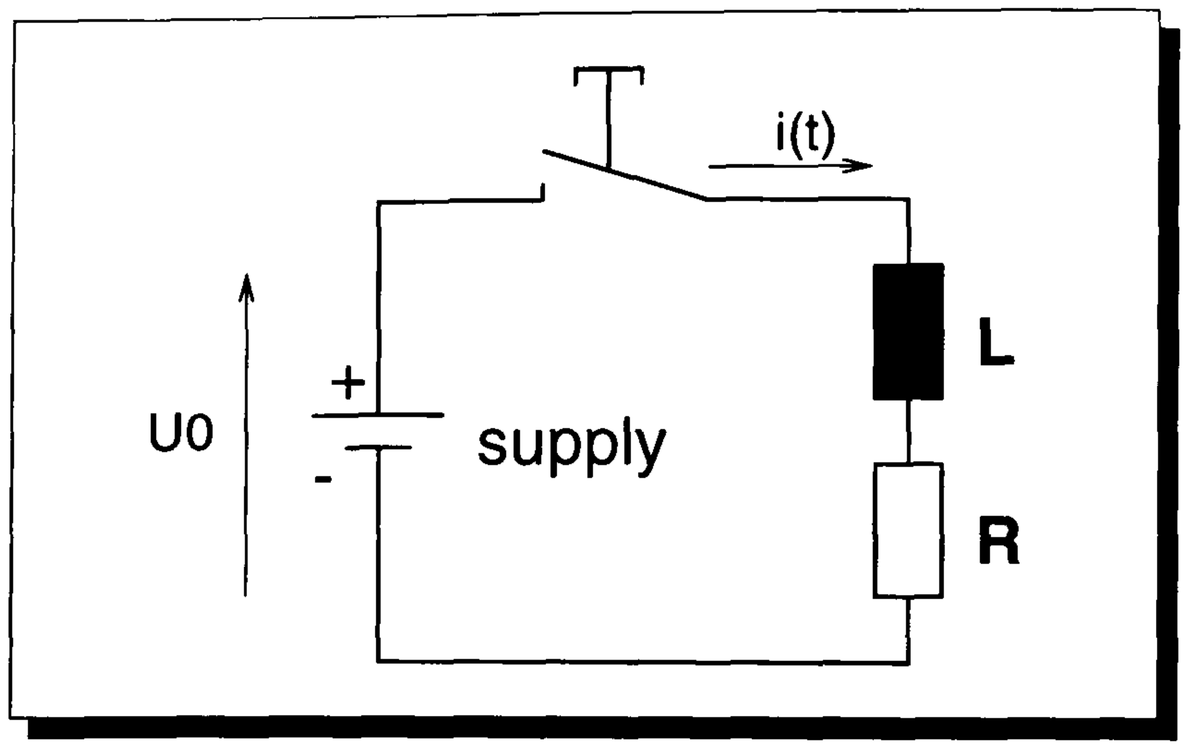


Figure 6.10: RL Series Circuit

Fig. 6.10 displays a typical 1st-order electrical system. The analogue circuit connects a resistor and an inductor in series supplied by a battery. After closing the switch at $t = 0$, the current can flow through the components. Suppose the current flow is of interest. The differential equation for this specific circuit is:

$$L * i'(t) + R * i(t) = U_0$$

Given the fuzzy values for:

$$R = (100, 10, 10)\Omega,\ L = (1, 0.1, 0.1)H,\ U_0 = (10, 1, 1)V$$

and the initial fuzzy values (at $t = 0$):

$$i(0) = (0, 0, 0)A \text{ (exactly known)}$$

The first order differential equation is represented by a chromosome as:

R	L	U_o	$i(0)$

The *maximum change parameter* for the particular values at the $\alpha - cut - level \geq 0.0$ are:

$$\begin{aligned} R &= [-10, 10]\Omega \\ L &= [-0.1, 0.1]H \\ U_0 &= [-1, 1]V \\ i(0) &= [0, 0]A \end{aligned}$$

The settings for the *Interactive Evolutionary Algorithm* for solving the first-order differential equation at $\alpha - cut - level \geq 0.0$ are:

Population size: 10 chromosomes

Mutation Rate: 3

At each generation the parameter set of the chromosomes are evaluated using the Runge-Kutta-Method. Runge-Kutta-Method settings:

Start Value: 0.000s

Stop Value: 0.600s

Step Size: 0.005s

After a simulation searching for the minimum curve the following tabular shows the best chromosome per generation with the new adapted mutation strength:

Generation	Mutation Strength (%)	R (Ω)	L (H)	U_0 (V)	$i(0)$ (A)
0	5	100.0	1.00	10.0	0
1	7	102.2	0.92	9.8	0
2	9	102.2	0.92	9.7	0
3	11	102.2	0.92	9.7	0
4	13	103.1	0.91	9.7	0
5	15	103.1	0.90	9.7	0
6	17	101.8	0.90	9.5	0
7	19	103.3	0.90	9.5	0
8	21	103.3	0.90	9.2	0
9	19	103.3	0.90	9.0	0
10	21	103.3	0.90	9.0	0
11	23	103.3	0.90	9.0	0
12	25	106.3	0.91	9.0	0
13	27	109.5	0.91	9.0	0
14	29	110.0	0.91	9.0	0

Generation	Mutation Strength (%)	R (Ω)	L (H)	U_0 (V)	$i(0)$ (A)
15	31	110.0	0.90	9.0	0
16	29	110.0	0.90	9.0	0
17	27	110.0	0.90	9.0	0
18	25	110.0	0.90	9.0	0
19	23	110.0	0.90	9.0	0

At each generation the Runge-Kutta-Method evaluates each chromosome and stores the generated x-y values in the database when they are better then the stored ones. The total result after the simulation is shown in Fig. 6.11.

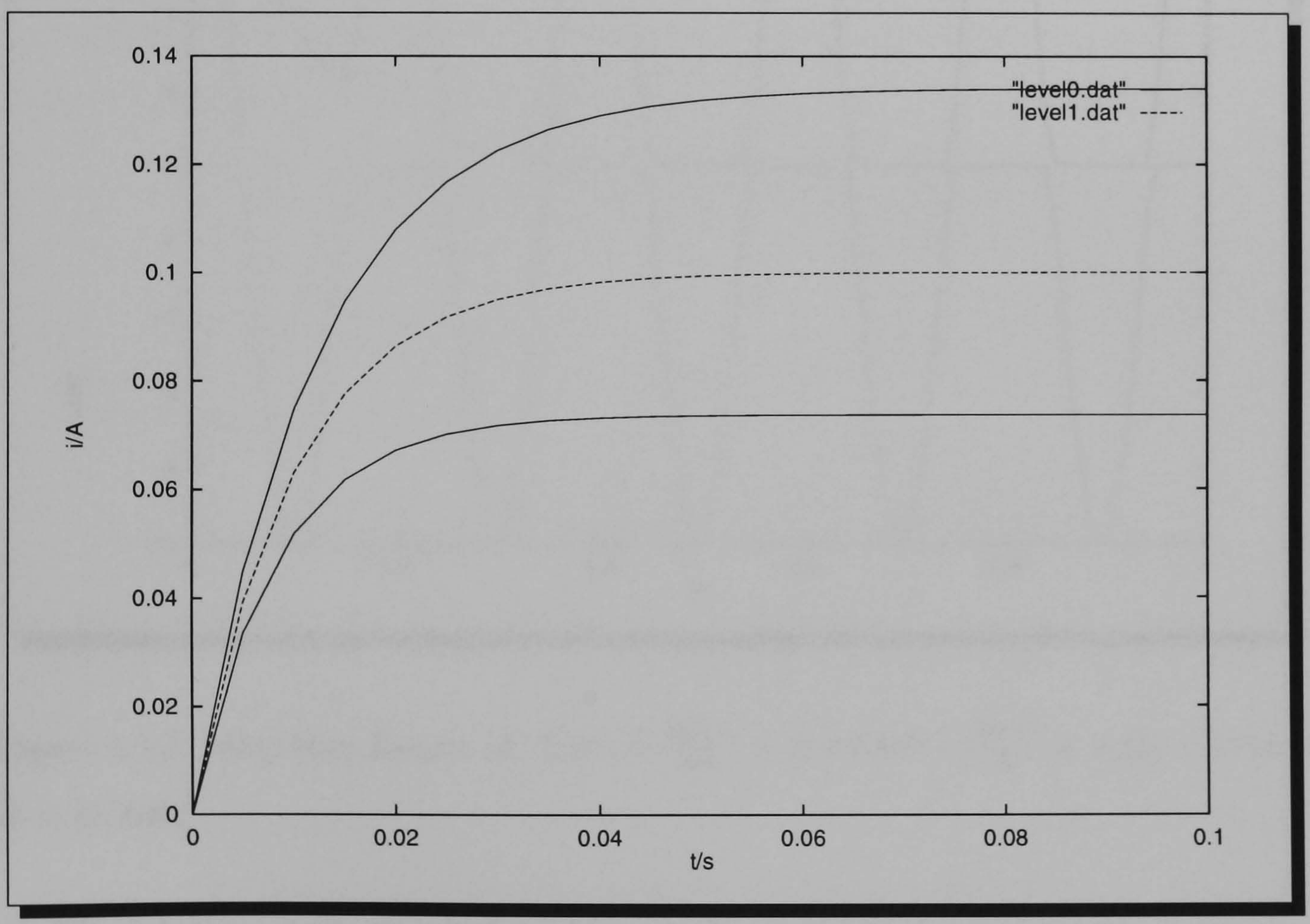


Figure 6.11: LR Series Circuit Simulation Result

6.7 Example: 2nd-Order Differential Equation

The result of the *Interactive Evolutionary Algorithm Approach* for the differential equation (discussed in Chapter 6.4)

$$0.001 * \frac{du_c^2(t)}{dt^2} + R * 0.001 * \frac{du_c(t)}{dt} + u_c(t) = 0$$

varying the resistor value between $R = 0\Omega$ and $R = 100\Omega$ is shown in Fig. 6.12.

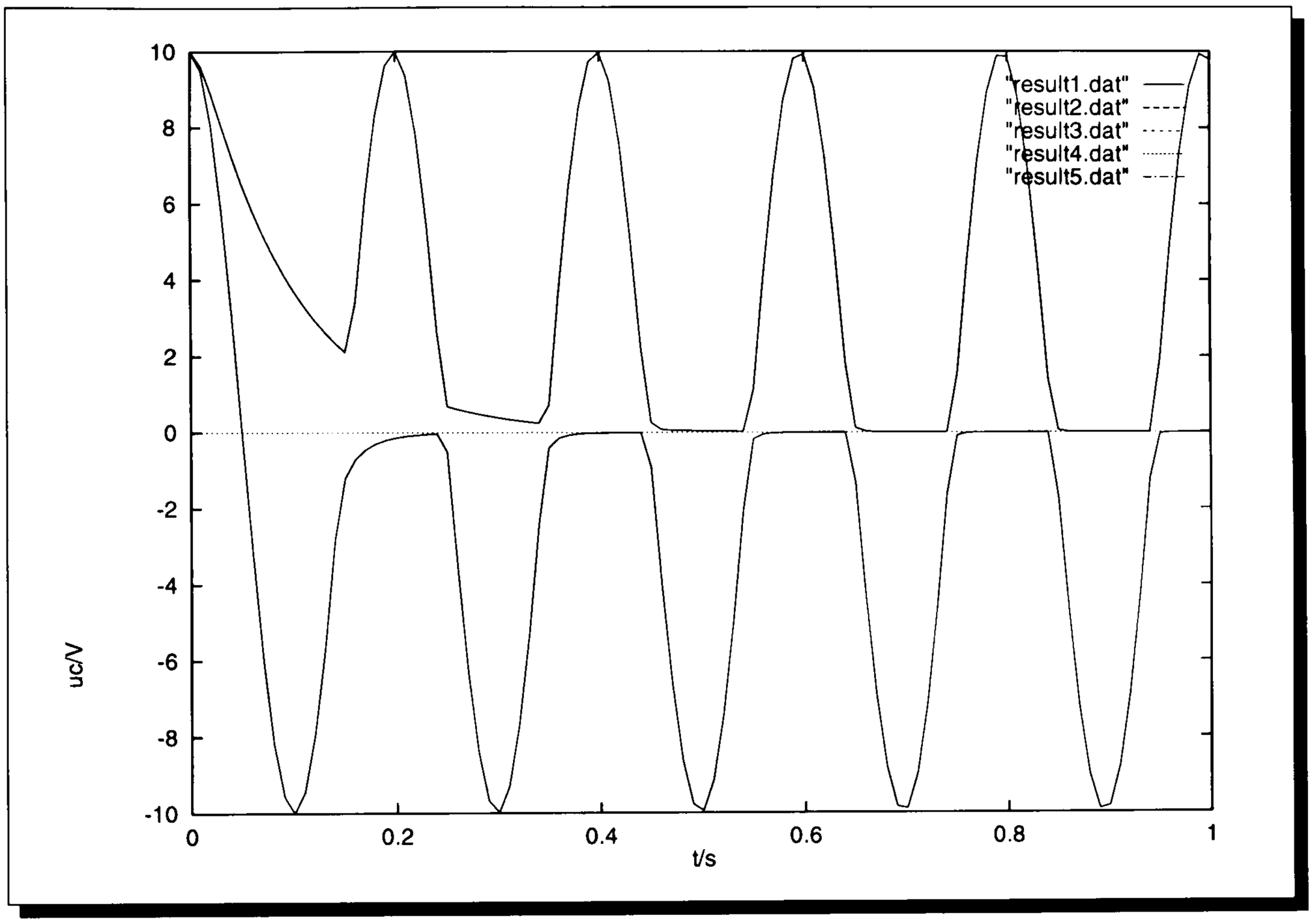


Figure 6.12: Min/Max Result of: $0.001 * \frac{du_c^2(t)}{dt^2} + R * 0.001 * \frac{du_c(t)}{dt} + u_c(t) = 0$ with $R = [0, 100]$

The solution of the simulation shown in Fig. 6.12 states the desired result discussed in Chapter 6.4. When several solutions have to be overlayed to get the overall extreme boundaries this approach is jumping between these solutions back and forth saving the results in the minimum/maximum database. When only one solution of

the differential equation represents the extreme boundaries, the evolutionary algorithm does not alternate between solutions and finds directly the optimum problem sets.

6.8 Example: 2nd-Order Differential Equation

The following 2nd-order differential equation has to be solved:

$$a2 * y'' + a1 * y' + a0 * y = 0$$

A typical 2nd-order differential electrical system is the series-connection of a resistor, an inductor, and a capacitor as shown in Fig. 6.1. At the beginning a switch prevents the capacitor to be discharged. Of interest is the voltage across the capacitor. The differential equation for the specific circuit is:

$$L * C * u_c''(t) + R * C * u_c'(t) + 1 * u_c(t) = 0$$

6.8.1 Aperiodic Case:

Given the fuzzy values for:

$$R = (100, 10, 10)\Omega,\ L = (1, 0.1, 0.1)H,\ C = (1, 0.1, 0.1)mF$$

and the initial fuzzy values (at $t = 0$):

$$u_c(0) = (10, 1, 1)V,\ u_c'(0) = 0V/s \text{ (exactly known)}$$

The result of the simulation is an aperiodic signal is shown in Fig. 6.13.

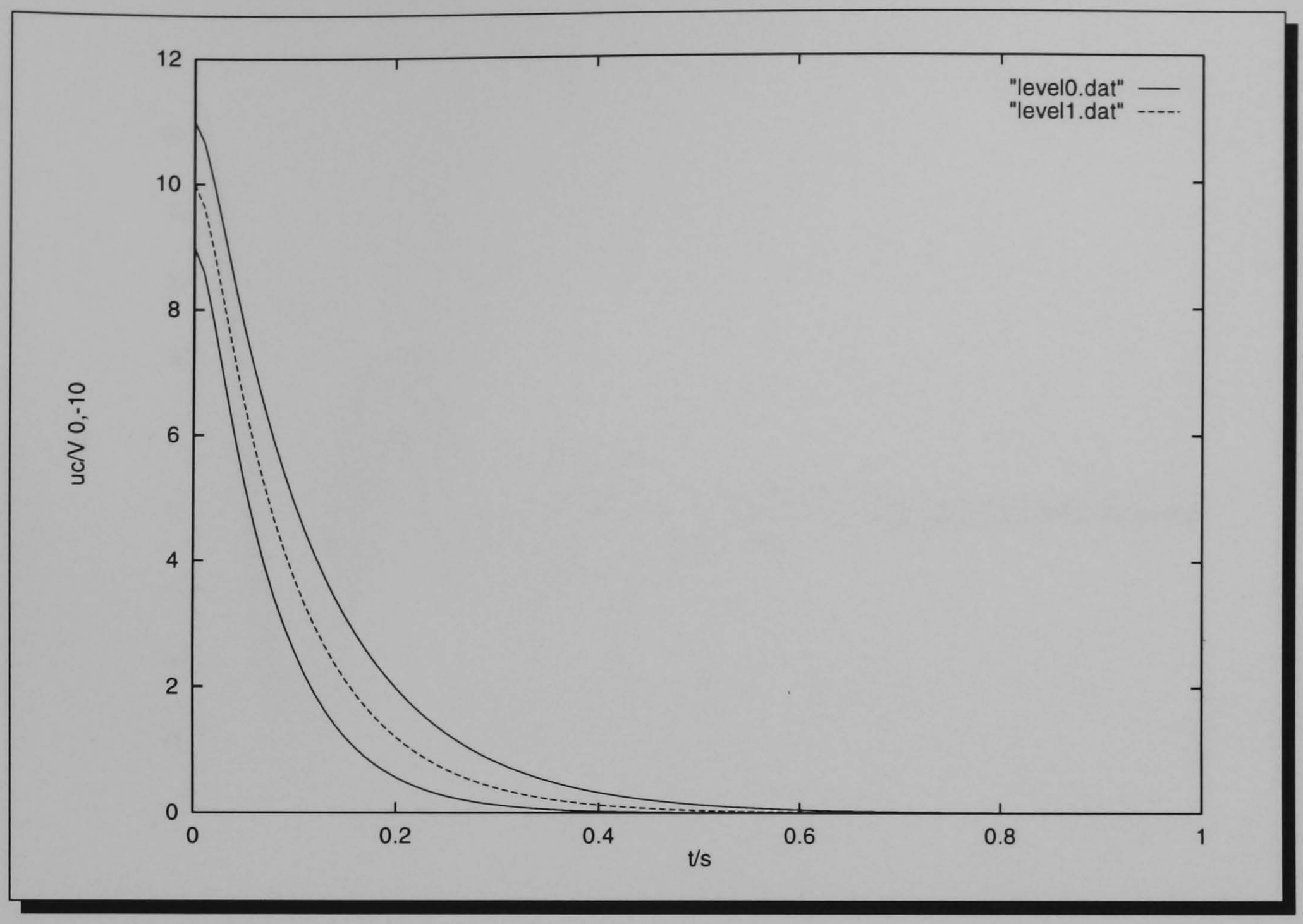


Figure 6.13: Aperiodic Case: LCR Circuit Simulation Result

6.8.2 Periodic Case With Attenuation:

Given the fuzzy values for:

$$R = (10.5, 0.5, 0.5)\Omega,\ L = (1, 0.1, 0.1)H,\ C = (1, 0.1, 0.1)mF$$

and the initial fuzzy values (at $t = 0$):

$$u_c(0) = (10, 1, 1)V,\ u_c'(0) = 0V/s \text{ (exact known)},$$

The result of the simulation is an periodic signal shown in Fig. 6.14. The signal is varying in amplitude and in frequency.

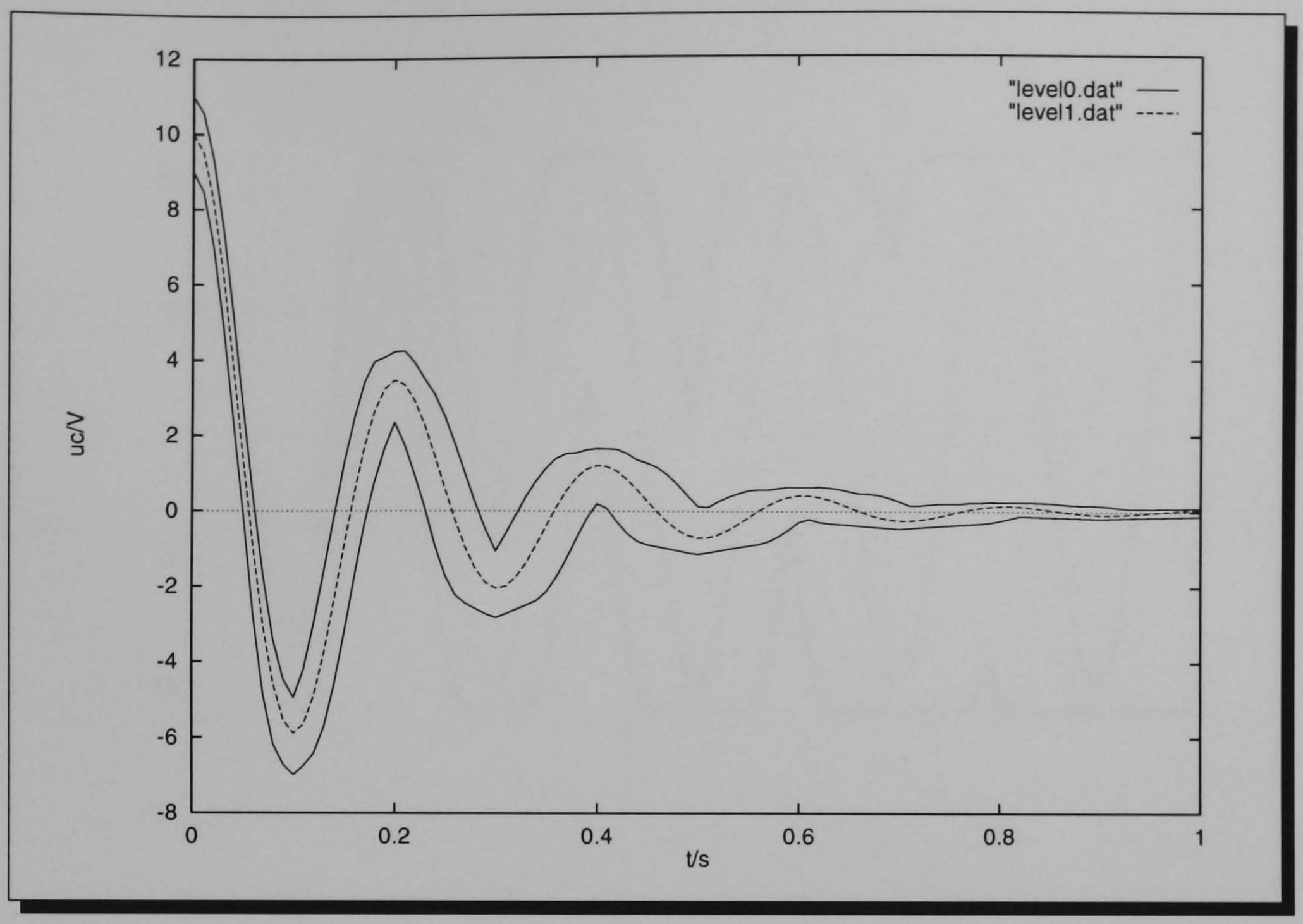


Figure 6.14: Periodic Case With Attenuation: LCR Circuit Simulation Result

6.8.3 Periodic Case Without Attenuation:

This is the case when the resistor is zero. Given the fuzzy values for:

$$R = (0, 0, 0)\Omega,\ L = (1, 0.1, 0.1)H,\ C = (1, 0.1, 0.1)mF$$

and the initial fuzzy values (at $t = 0$):

$$u_c(0) = (10, 1, 1)V,\ u_c'(0) = 0V/s \text{ (exact known)},$$

The result of the simulation is an periodic signal and shown in Fig. 6.15.

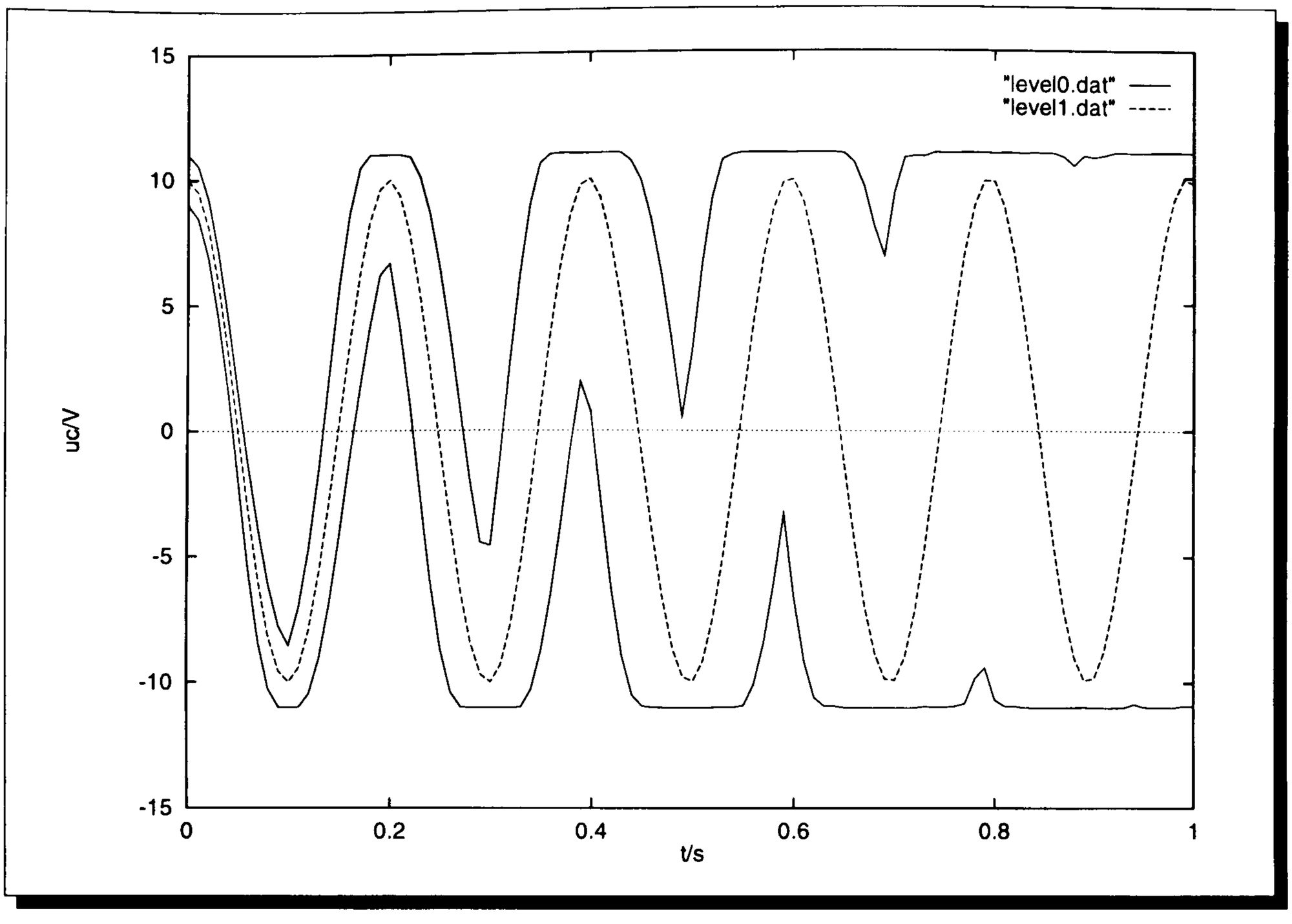


Figure 6.15: Periodic Case Without Attenuation: LCR Circuit Simulation Result

6.9 Conclusion

Solving differential equations is difficult even when there is no imprecision involved. Introducing imprecision makes it very hard or even impossible to solve the problem correctly. The approach introduced in this chapter is a mixture between the interacting and non interacting approach of Bonarini and Bontempi [Bonarini and Bontempi, 1994]. The *Interactive Evolutionary Algorithm Approach* tries to find the overall maximal and minimal curves at each α-cut level. It is based on the classical evolutionary algorithm using the protocol system that saves the minimum/maximum values during the process in a minimum/maximum database. The drawback of this approach is the high number of evaluations.

Chapter 7

Imprecise Modelling

This chapter discusses functional-based, rule-based, and constraint-based modelling of dynamic systems not known exactly. It investigates fuzzy rule-based modelling, often called fuzzy nonlinear systems, in detail because the *Fuzzy Relation Memories* (defined in Chapter 4) are a subset of such modelling approach. A *Fuzzy Relation Memory* example and a conclusion summarize this chapter.

7.1 Representation of Structure

A human perception of the system is based on experience, expertise, empirical observations, intuition, a knowledge of the physical properties of the system, or a set of subjective preferences and goals. This type of knowledge is preferred by human observers modelling systems. As described in [Lunze, 1995], there are two techniques modelling dynamic systems:

- by sets of data and IF-THEN rules. With IF-THEN rules, input/output data sets are related. This representation is often called rule-based modelling or shallow modelling.

- by model equations. This compact description allows for any input to evaluate an output value, also called deep modelling.

Using the IF-THEN rules and equations three types of dynamic models have been extracted as formulated by Fishwick [Fishwick, 1997]:

1. **Declarative Models:** States and events are the focus of declarative models. IF-THEN rules are a possible approach for such models. Dynamic modelling by rule-based systems is more applicable for novice users. It is not necessary to have a deep understanding of the modelling structure.

2. **Functional Models:** Designing at the block level model the functions are the main components of a functional model. These blocks are constructed rule-based and/or constraint-based.

3. **Constraint Models:** They are represented by sets of equations or as constraint networks. These models require knowledge, called deep knowledge, of the internal functionality and structure of the system.

Depending on the purpose of the modelling task, a specific type of modelling is used. The level of detail of a model depends highly on the purpose of the modelling tasks [de Kleer, 1977]. For studying car design the model of, for example, a tyre is far less detailed then for roll sound investigations.

Three kinds of modelling, as described in the system MOOSE (Multi model Object Oriented Simulation Environment) by Cubert and Fishwick [Cubert and Fishwick, 1997], are used in this thesis for modelling dynamic systems and related to imprecision:

1. **imprecise equation-based** modelling (see Section 7.2),

2. **imprecise functional-based** block modelling (see Section 7.3), and

3. **imprecise rule-based** modelling (see Section 7.4).

All three modelling approaches can handle imprecision through their imprecise defined parameters. Therefore the modelling notations are extended by the word "imprecise".

7.2 Imprecise Equation-Based Modelling

Imprecise equation based modelling has a number of n-th order differential equations. Differential equations are represented using symbols such as x, x' for the first derivative of x, x" for the second derivative of x, and in general x followed by n single quote marks to denote the nth derivate of x. Fig. 7.1 displays the frictionless spring problem:

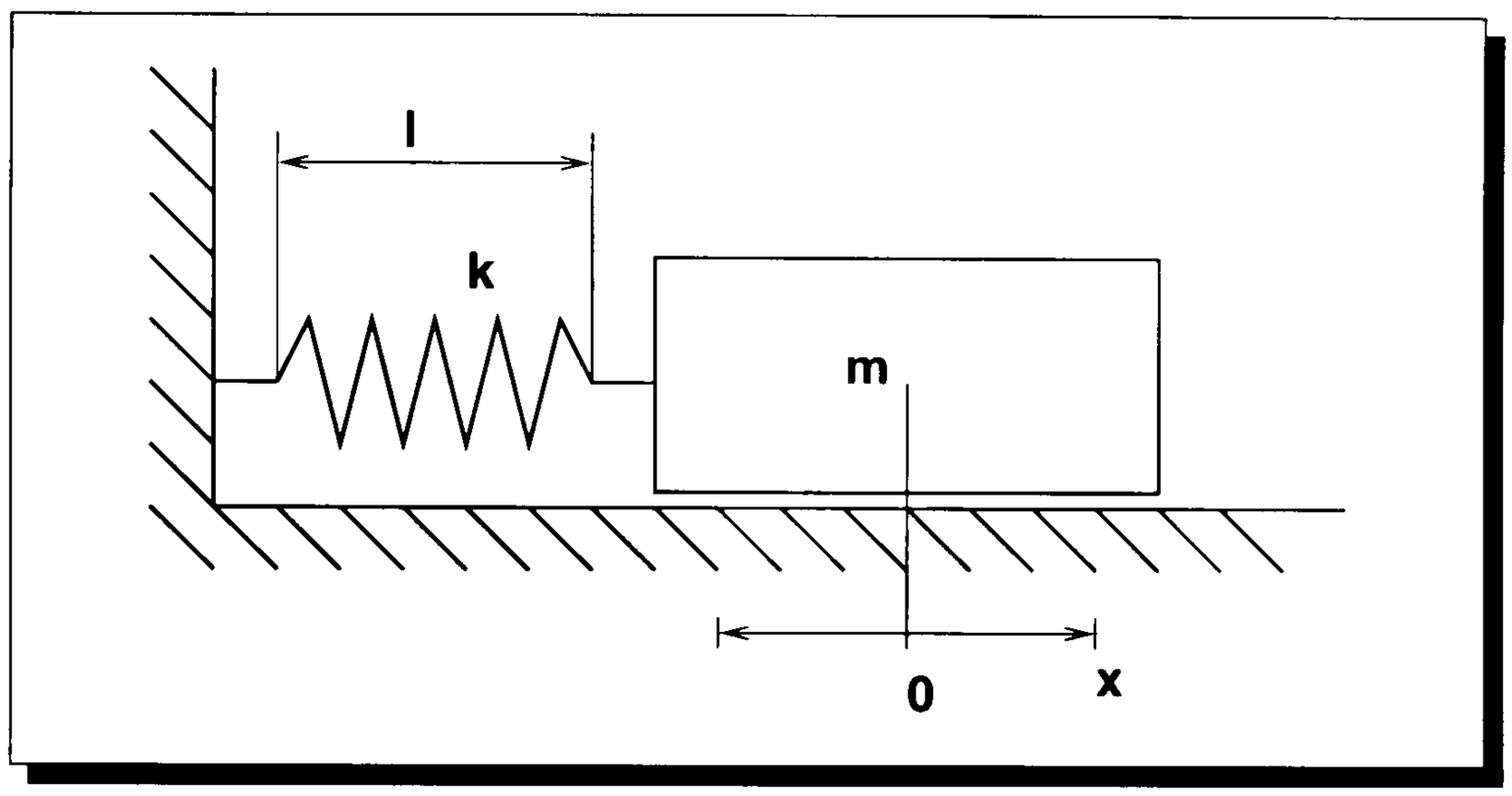


Figure 7.1: Equation Based Modelling of a Resistor

x is the position of the mass m. k is a factor calculated on the basis of the mass and the natural length l of the spring. The frictionless spring problem could be equation-based modelled by the following differential equation:

$$m * \frac{d^2x}{dt^2} + k * x = 0$$

If the parameters or the initial values are not known exactly the equation is an imprecise differential equation and solved by the *Interactive Evolutionary Algorithm Approach* as discussed in Chapter 6.

7.3 Imprecise Functional-Based Modelling

The basis of imprecise functional block modelling is a block with imprecise parameters specifying the block. Blocks represent particular functions e.g. multiply, add, divide, integrate, etc. Several basic functional blocks are connected; no functional block is isolated. Cycles are permitted, in which case the value at one time step propagates to the next time step. Fig. 7.2 shows an ideal differential amplifier stage without limitation of the output (theoretical infinite output value).

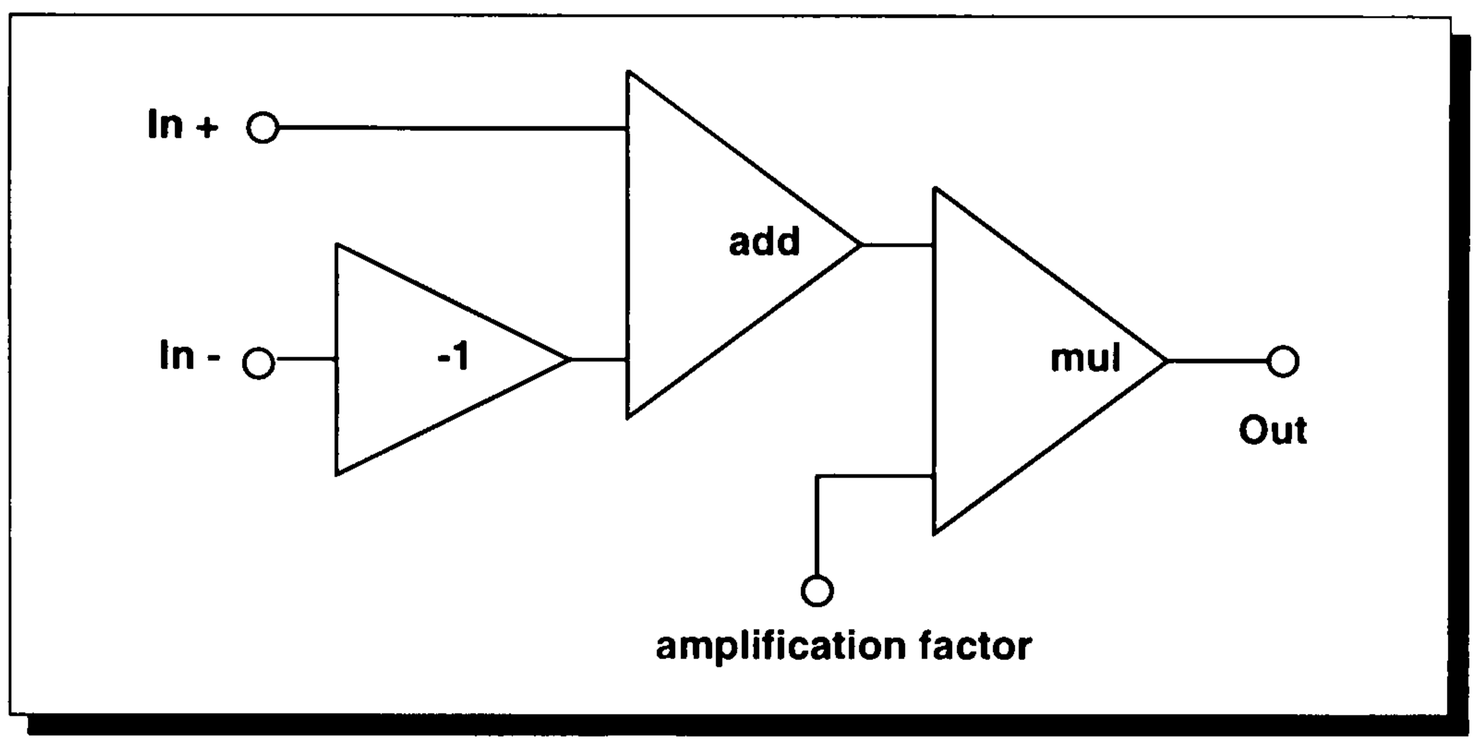


Figure 7.2: Functional-Based Modelling of a Differential Amplifier Stage

As described in Chapter 5 blocks can be combined by ordinary mathematical operations.

7.4 Imprecise Rule-Based Modelling

Fuzzy rule-based models, also called fuzzy systems or fuzzy models, have in common that they describe a nonlinear dynamic system by discrete relations of linguistic information (see [Tong, 1979] in [Gupta et al., 1979]). In modelling nonlinear systems, five types of fuzzy rule-based systems are considered. Some of them are discussed in Vadiee's Chapter: "Fuzzy rule-based expert systems — I and II" in the book [Jamshidi et al., 1993]. These five fuzzy systems discussed are described by a col-

lection of r restrictions in the form of IF-THEN rules, shown in Fig. 7.3:

$$\begin{aligned}
\tilde{R}_1&: \quad \text{IF } x \text{ is } \tilde{A}_1, \text{ THEN } y \text{ is } \tilde{B}_1 \\
\tilde{R}_2&: \quad \text{IF } x \text{ is } \tilde{A}_2, \text{ THEN } y \text{ is } \tilde{B}_2 \\
&\cdot \qquad\qquad\qquad \cdot \\
&\cdot \qquad\qquad\qquad \cdot \\
&\cdot \qquad\qquad\qquad \cdot \\
\tilde{R}_r&: \quad \text{IF } x \text{ is } \tilde{A}_r, \text{ THEN } y \text{ is } \tilde{B}_r
\end{aligned}$$

Figure 7.3: Canonical Rule-Based Form of Fuzzy Relational Equations

Fuzzy Relation Memories, as discussed in Chapter 4, belong to the 5th type. They also consist of the canonical rule-based form with the difference that fuzzifying functions $\tilde{f}_r(x)$ are the consequence of the IF-THEN rules as shown Table 4.7 in Chapter 4.

The five types are for single-input and single-output systems described, but the results can be easily extended to n-input and m-output systems where input $\mathbf{x}$ is an n-dimensional vector and output $\mathbf{y}$ is an m-dimensional vector. All types represent special restrictions on the antecedents (input) and the consequents (output).

7.4.1 Type I: Singleton Input; Singleton Output

In the first type of fuzzy model, the input and the output restrictions are given in the form of singletons, i.e., $\tilde{A}_1 : x = x_1$, $\tilde{A}_2 : x = x_2$, ..., and $\tilde{B}_1 : y = y_1$, $\tilde{B}_2 : y = y_2$ .... This type is simply a lookup table for the system description visualized by Fig. 7.4:

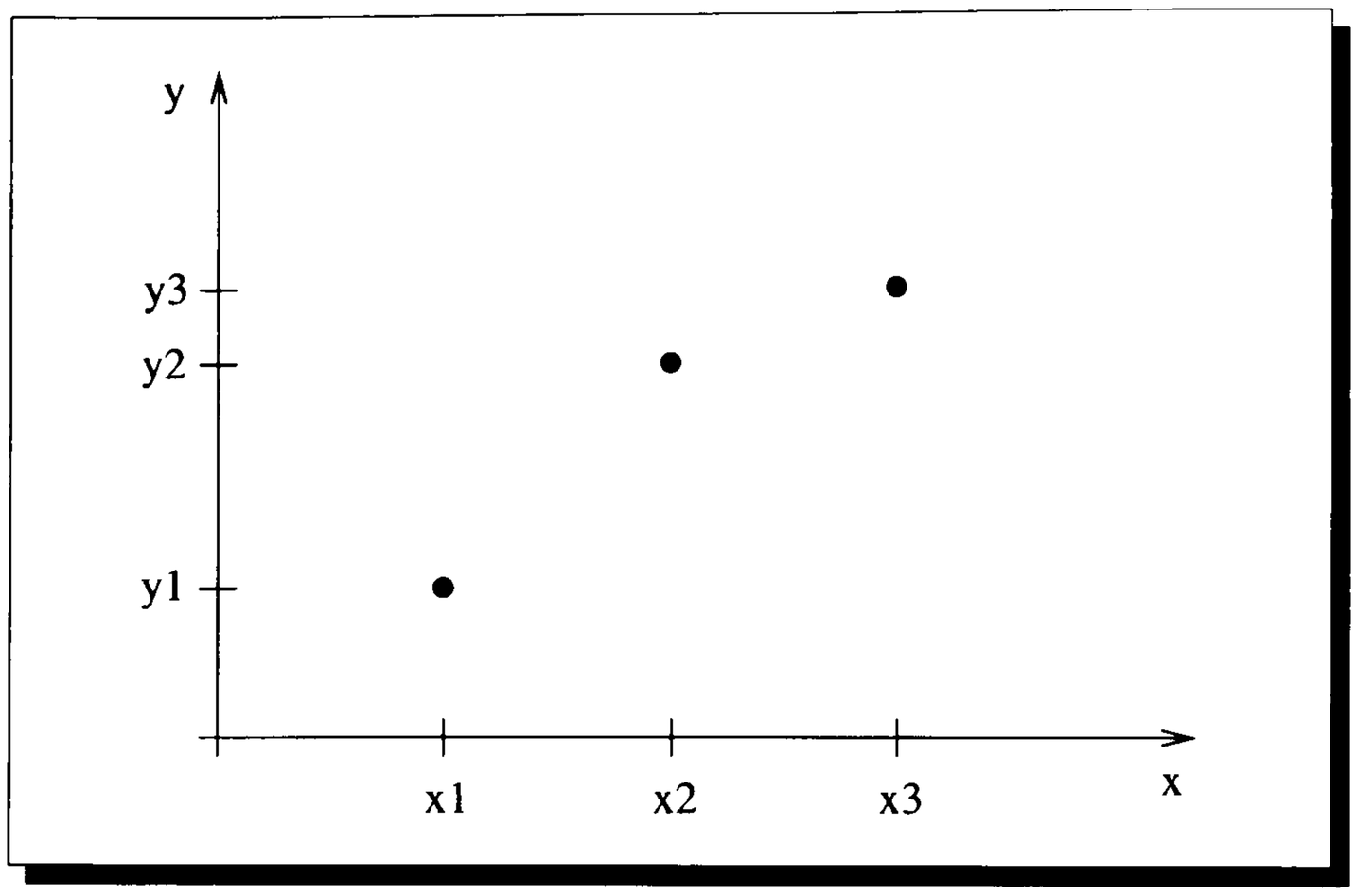


Figure 7.4: Input Singletons and Output Singletons

Equation given for Fig. 7.4:

$$\text{IF } \tilde{A}_i : x = x_i \text{ THEN } \tilde{B}_i : y = y_i \quad \text{for i=1,2,...,r} \tag{7.1}$$

7.4.2 Type II: Crisp Interval Input; Singleton or Function Output

In the second type of fuzzy nonlinear models, the input restrictions are in the form of crisp sets and the output restrictions are given by singletons. This is also a lookup table for the system description. The value of the output for a given value of input is equal to the THEN part value of the ith rule $\tilde{R}_i$ whose IF part occurs somewhere in the interval, as seen in Fig. 7.5.

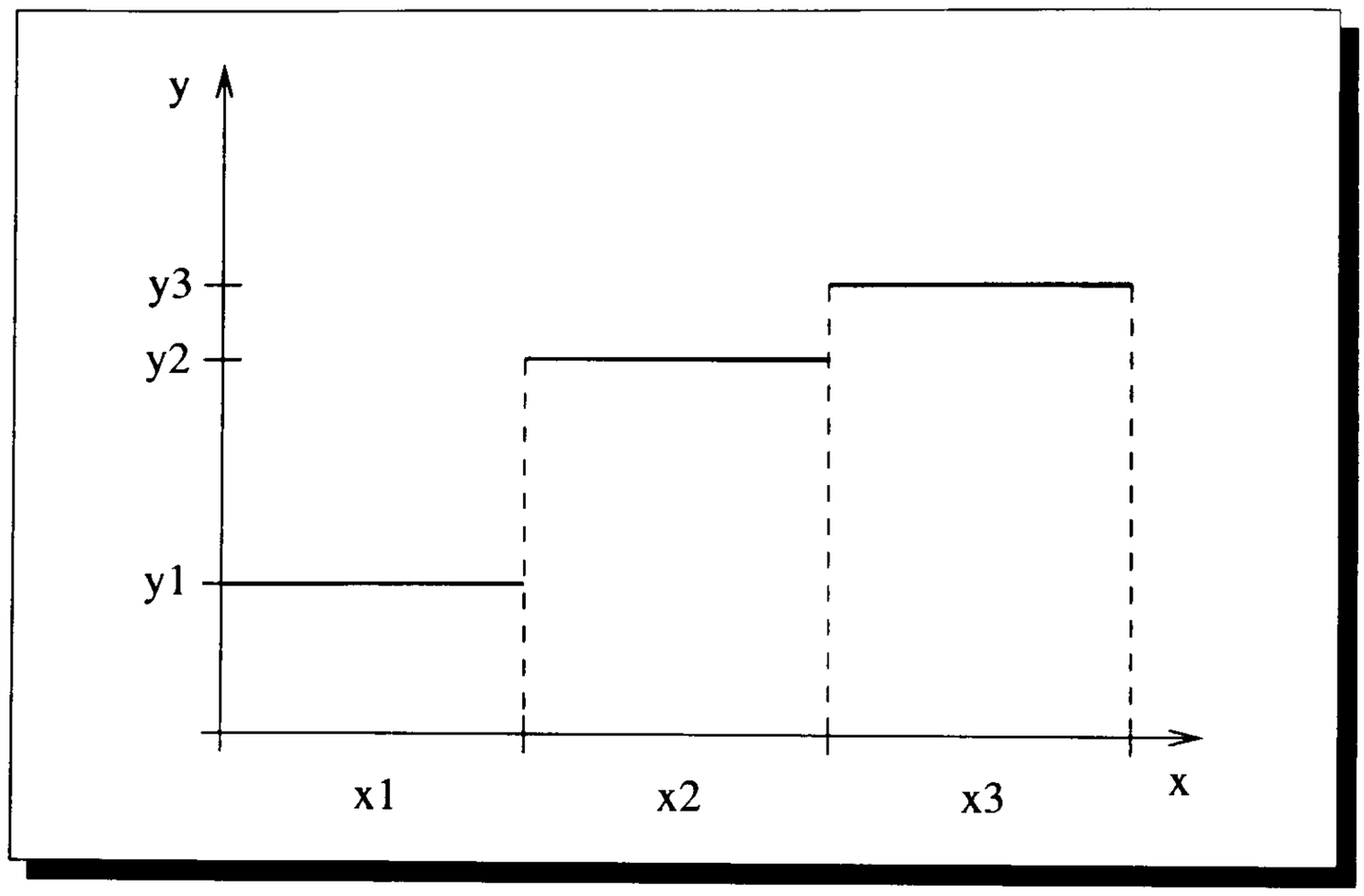


Figure 7.5: Input Crisp Intervals and Output Singletons

Equation given for Fig. 7.5:

$$\text{IF } \tilde{A}_i : x_{i-1} < x < x_i \text{ THEN } \tilde{B}_i : y = y_i \quad \text{for i=1,2,...,r} \tag{7.2}$$

This type is effectively a piecewise-constant approximation of a nonlinear function. The restrictions for the second type of fuzzy model might also involve spline functions to represent the output instead of crisp singletons. In this case the nonlinear function is represented by functions $f_1(x), f_2(x), ..., f_r(x)$, which are nonlinear spline functions (see Fig. 7.6).

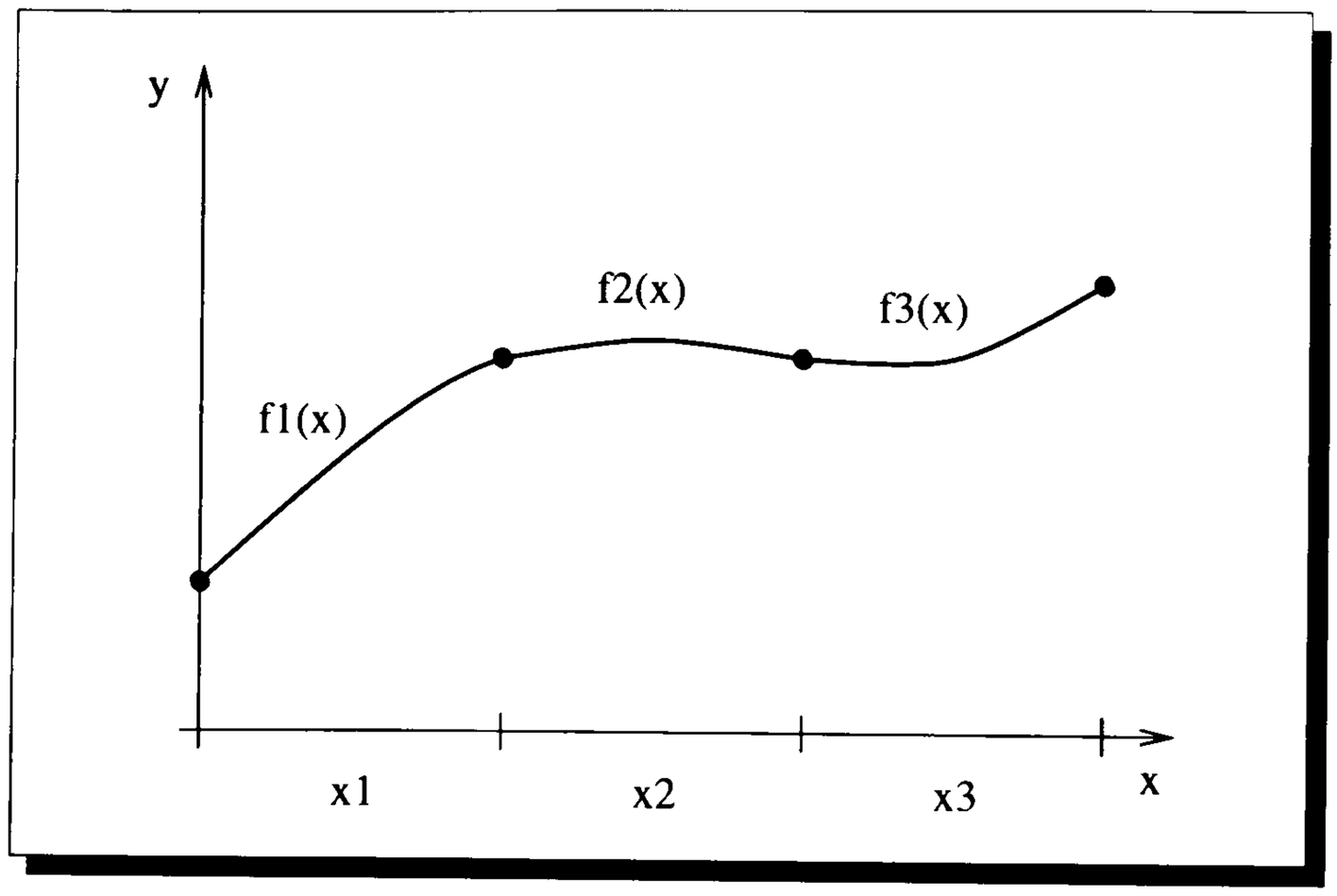


Figure 7.6: Input Crisp Intervals and Output Crisp Functions

Equation given for Fig. 7.6:

$$\text{IF } \tilde{A}_i : x_{i-1} < x < x_i \text{ THEN } \tilde{B}_i : y = f_i(x) \quad \text{for i=1,2,...,r} \tag{7.3}$$

7.4.3 Type III: Crisp Interval Input; Fuzzy Set Output

In the third class of fuzzy nonlinear models, the input conditions are crisp sets and the output is expressed as a fuzzy set or described by a fuzzy relation.

Equation given:

$$\text{IF } \tilde{A}_i : x_{i-1} < x < x_i \text{ THEN } \tilde{B}_i : y = \tilde{R}_i \quad \text{for i=1,2,...,r} \tag{7.4}$$

$\tilde{R}_i$ is either a fuzzy set or a fuzzy relation and can be defuzzified if necessary. In Fig. 7.7 the input space is divided into crisp intervals; the intervals may overlap. These inputs map to relations to the output.

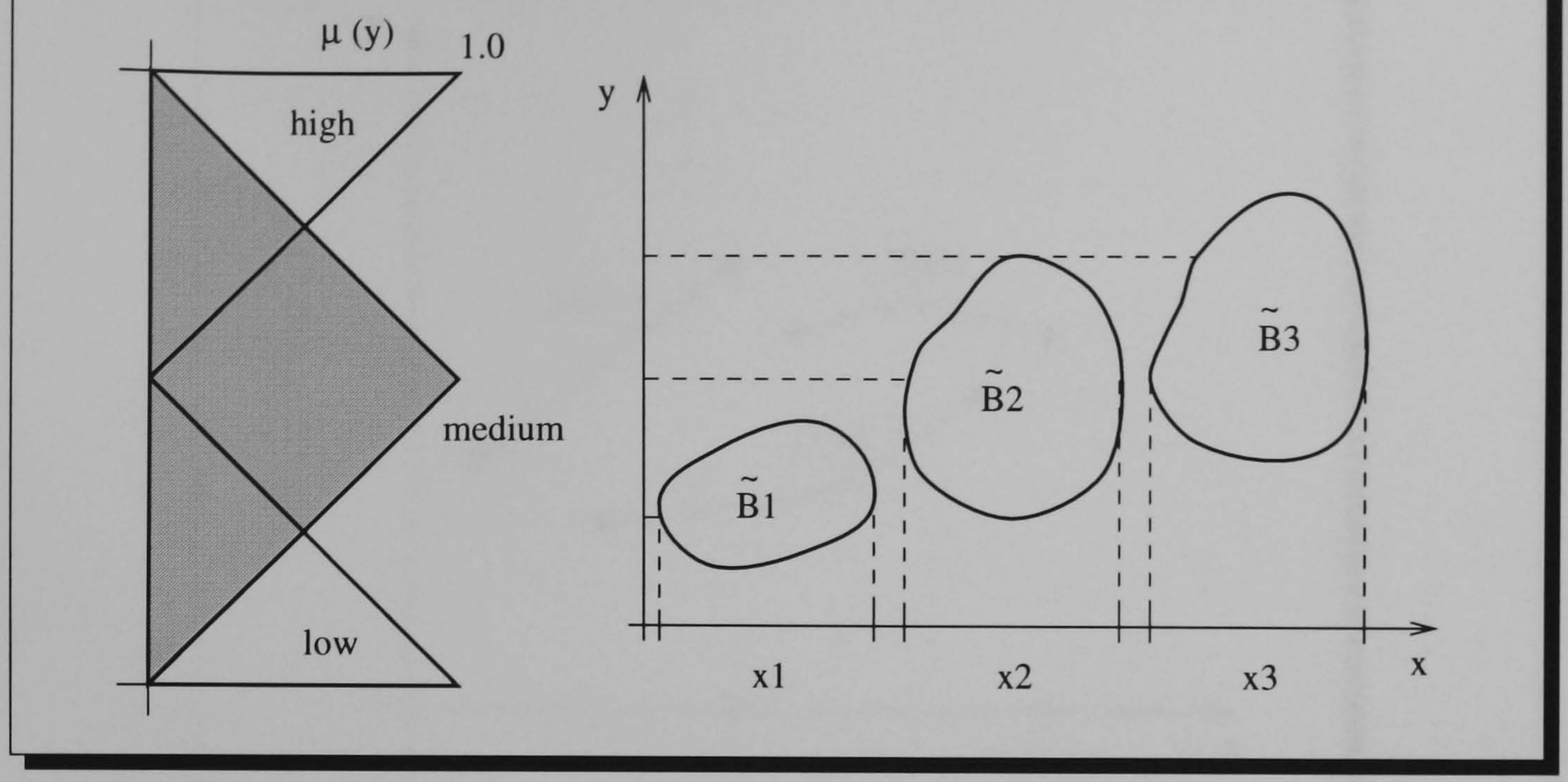


Figure 7.7: Input Crisp Intervals and Output Fuzzy Sets

7.4.4 Type IV: Fuzzy Set Input; Singleton or Function Output

In the fourth type of fuzzy models, the input conditions are given in the form of fuzzy sets, $\tilde{A}_i$, partitioned on the input universe of discourse, and the output are given in the form of singletons or generally nonlinear crisp functions.

Equation given for singletons:

$$\text{IF } x = \tilde{A}_i \text{ THEN } \tilde{B}_i : y = y_i \quad \text{for i=1,2,...,r} \tag{7.5}$$

Fig. 7.8 shows an example for a fourth type fuzzy model. The output is some weighted average of the spline functions.

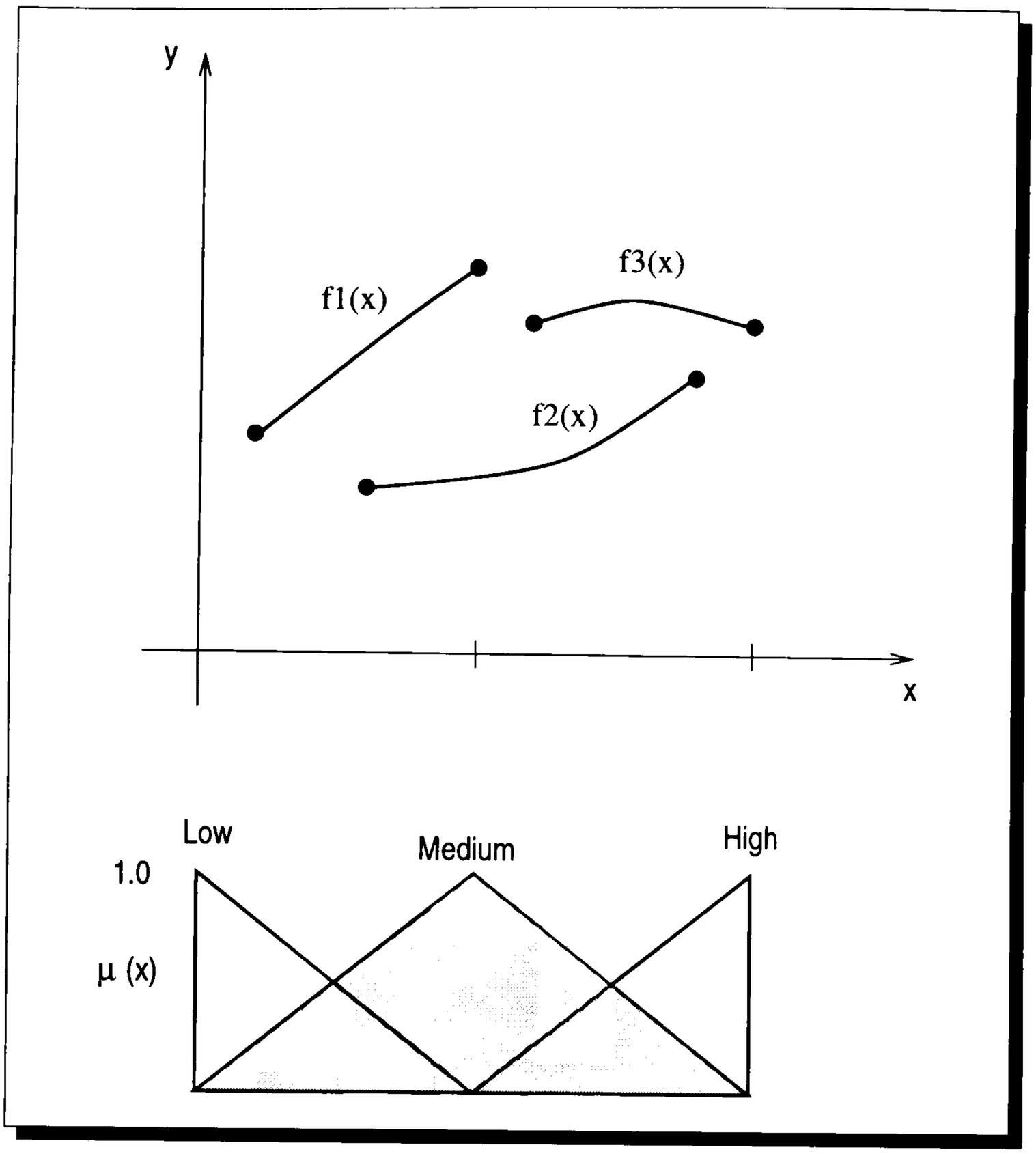


Figure 7.8: Input Fuzzy Sets and Output Crisp Functions

Equation given for functions:

$$\text{IF } x = \tilde{A}_i \text{ THEN } \tilde{B}_i : y = f_i(x) \quad \text{for i=1,2,...,r} \tag{7.6}$$

7.4.5 Type V: Fuzzy Set Input; Fuzzy Set Output

The most general fuzzy model for nonlinear simulation is given when both the input and the output restrictions are described by fuzzy sets shown in Fig. 7.9.

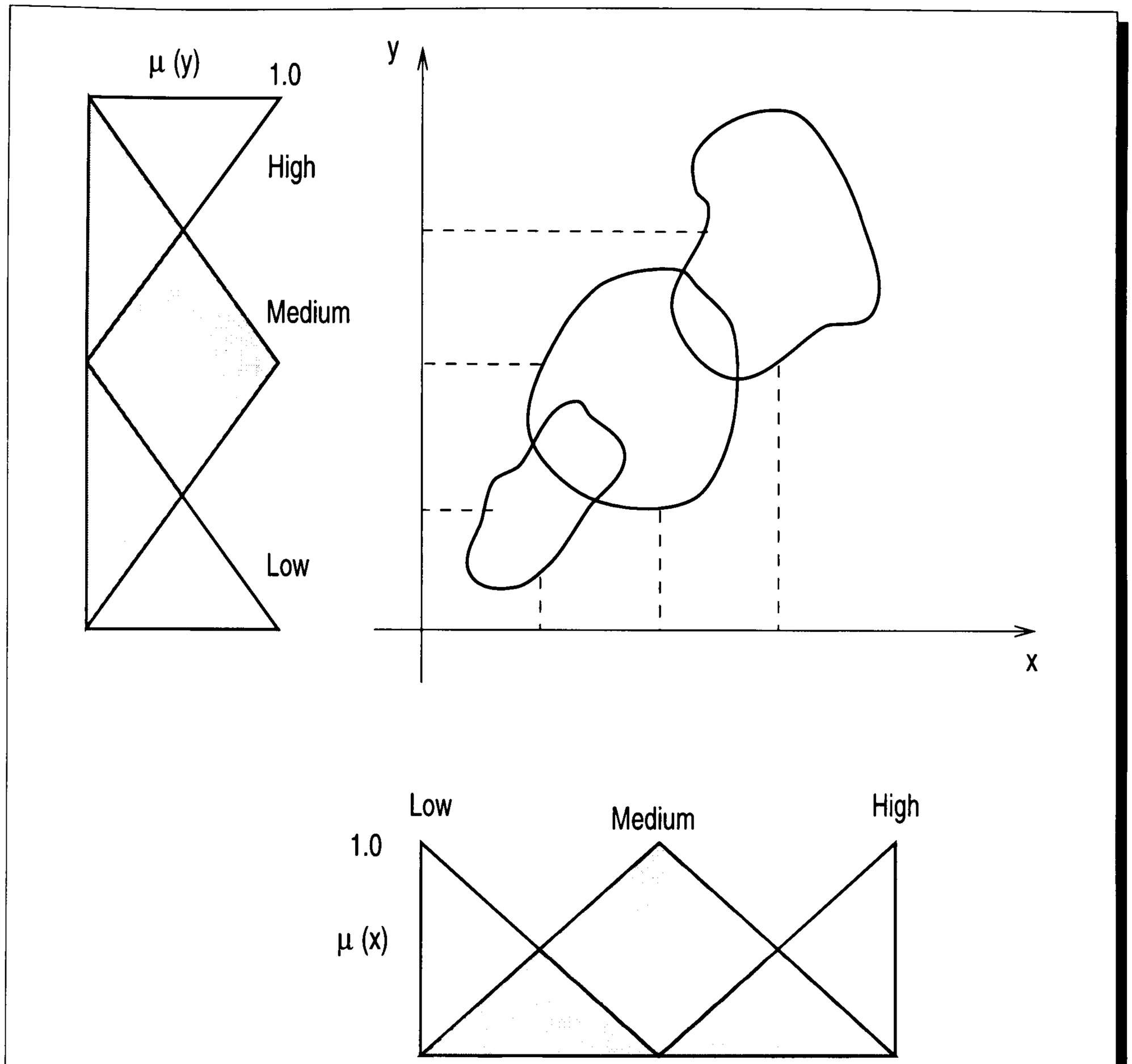


Figure 7.9: Input Fuzzy Sets and Output Fuzzy Sets

The Fuzzy Associative Memory (FAM), as described in Chapter 2, will be illustrated for a fuzzy system with 2 inputs (A and B) and 1 output (C). This small number of inputs make it possible to represent it in a tabular format as in the

following example:

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
B_1	C_1		C_4	C_4		C_3	C_3
B_2		C_1				C_2	
B_3	C_4		C_1			C_2	C_2
B_4	C_3	C_3		C_1		C_1	C_2
B_5	C_3		C_4	C_4	C_1		C_3

Figure 7.10: FAM Table for a Two-Input, Single-Output Fuzzy Rule-Based System

In Table 7.10 there are seven partitions for input A, five partitions for input B, and four partitions for the output variable C. This compact graphical form is called a fuzzy associative memory table, (FAM table). The FAM table maps patches of the input space to the patches in the output space.

A consequent extension of the FAMs are the Fuzzy Relation Memory (FRM) (defined in Chapter 4) that are fuzzifying functions in the THEN part, shown in Fig. 7.11.

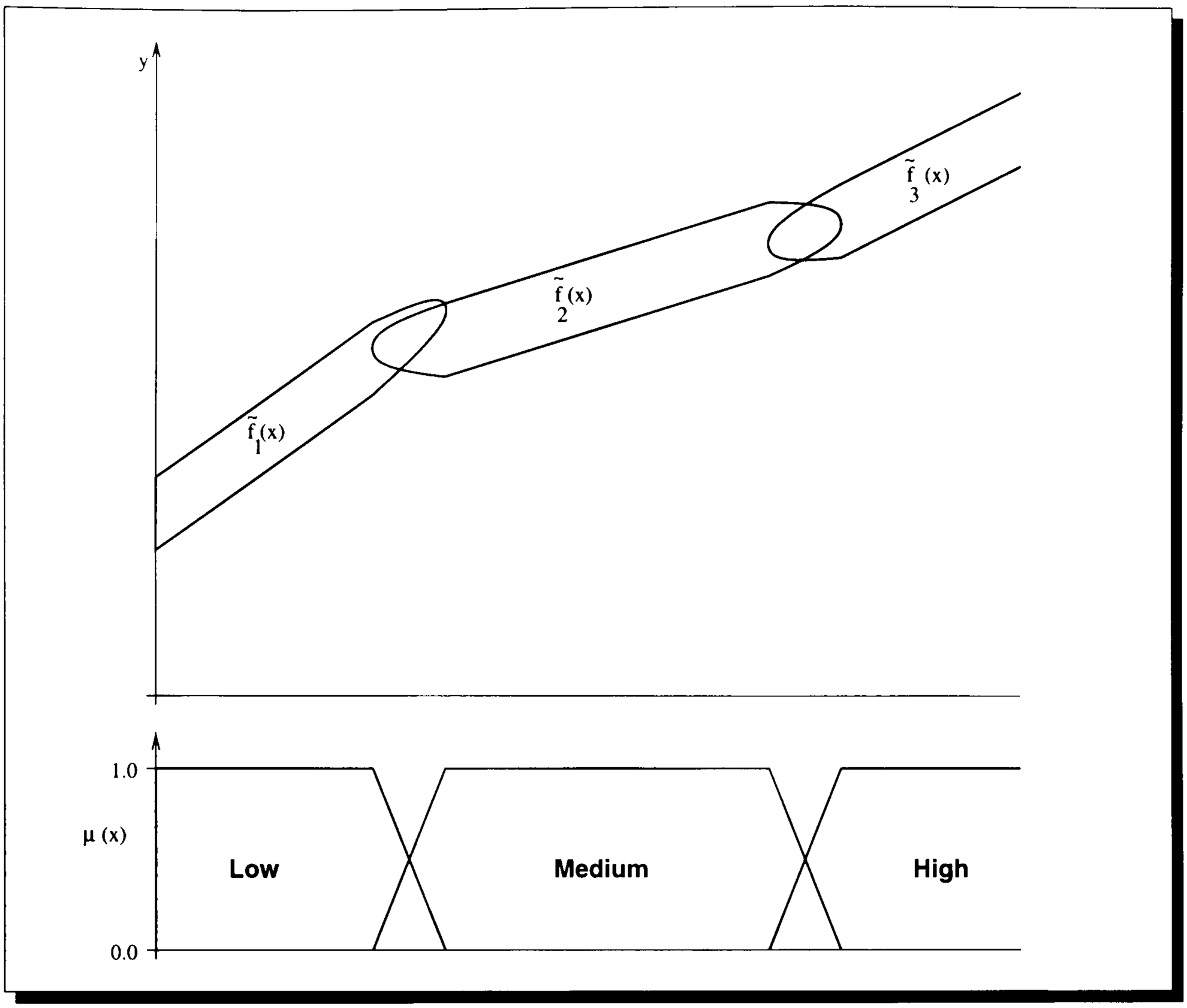


Figure 7.11: Input Fuzzy Sets and Output Fuzzifying Functions

The FRM maps patches of the input space to imprecise functions of the output space. FRM models can represent highly complex and nonlinear systems using rule-based modelling.

If linguistic variables are used the Fuzzy Associative Memory and the Fuzzy Relation Memory can be seen as qualitative modelling of a system.

FRM models can be combined to more complex models using algebraic operators as described in Chapter 5.

7.4.6 Example: Voltage Controlled Current Source

A simple operational amplifier circuit (Fig. 7.12) builds a voltage controlled current source.

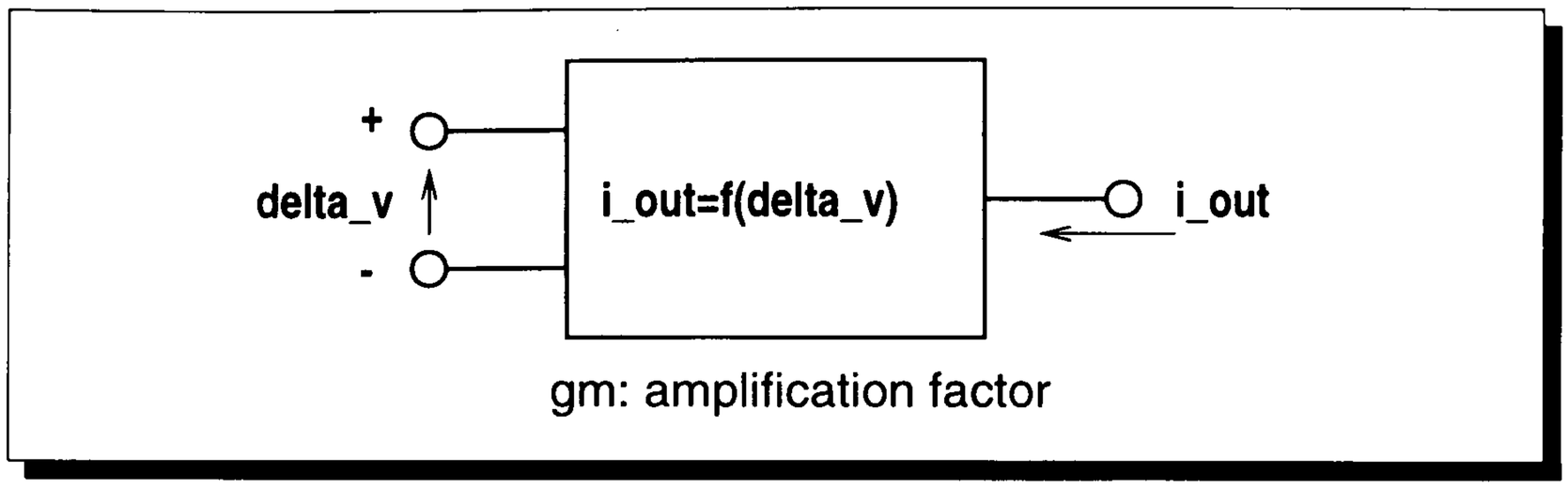


Figure 7.12: Operational Amplifier Block

The exact behaviour of the circuit is described by the following rules:

```
IF      (delta_v < -delta_v_max)   THEN   i_out = -i_max
IF  (delta_v >= -delta_v_max)&&
        (delta_v <= delta_v_max)   THEN   i_out = gm * delta_v
IF       (delta_v > delta_v_max)   THEN   i_out = i_max
```

Figure 7.13: Exact Rule-Based Modelling

Suppose an imprecise qualitative model is built. This circuit can be modelled by a fuzzy rule-based system with one-input `delta_v` and one-output `i_out`. The input space is portioned into the three fuzzy variables `below delta_v`,`delta_v`, and `above delta_v` as shown in Fig. 7.14.

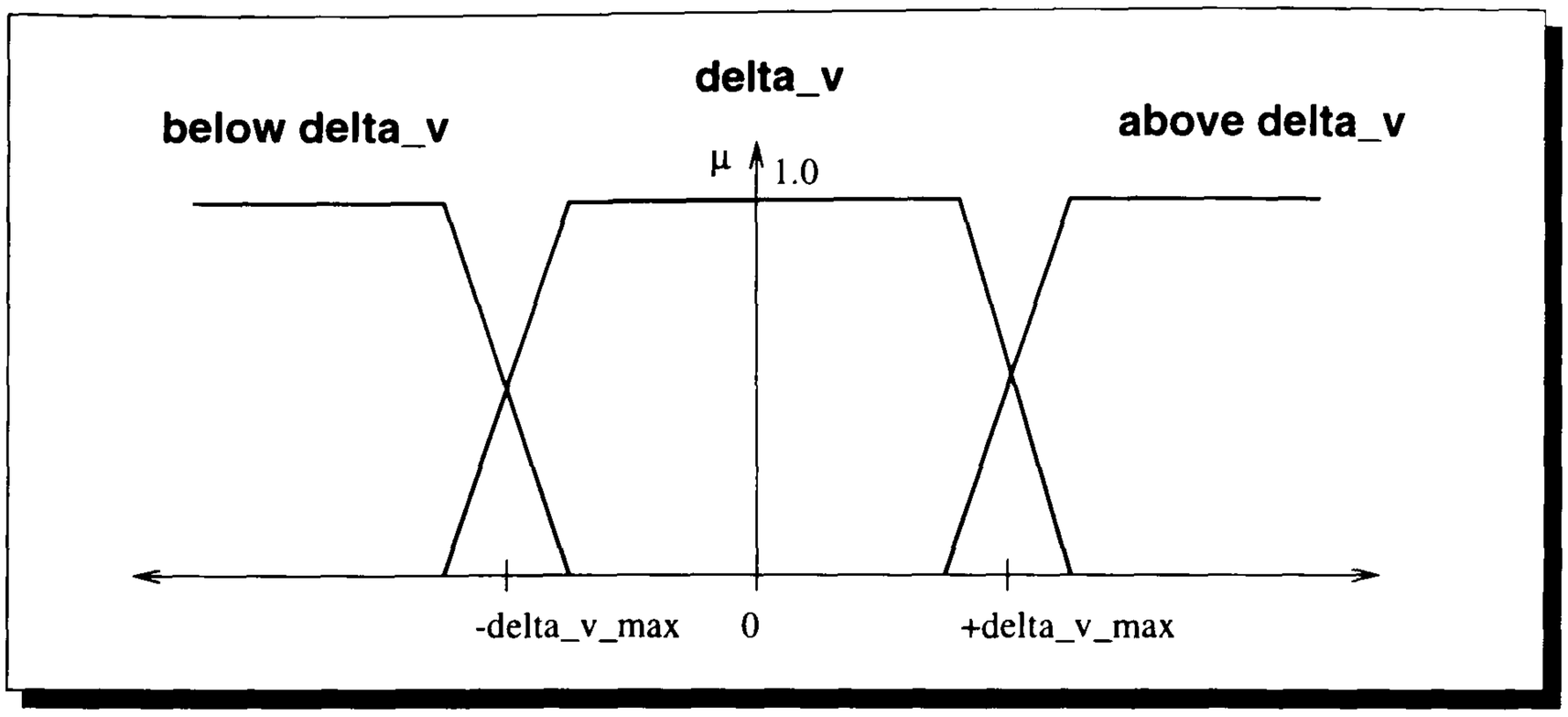


Figure 7.14: Three Partitions for the Input Variable

Given this input space a Fuzzy Relation Memory is used to model the nonlinear system. The defined input space is mapped with three IF-THEN rules to three fuzzifying functions as shown in Fig. 7.15:

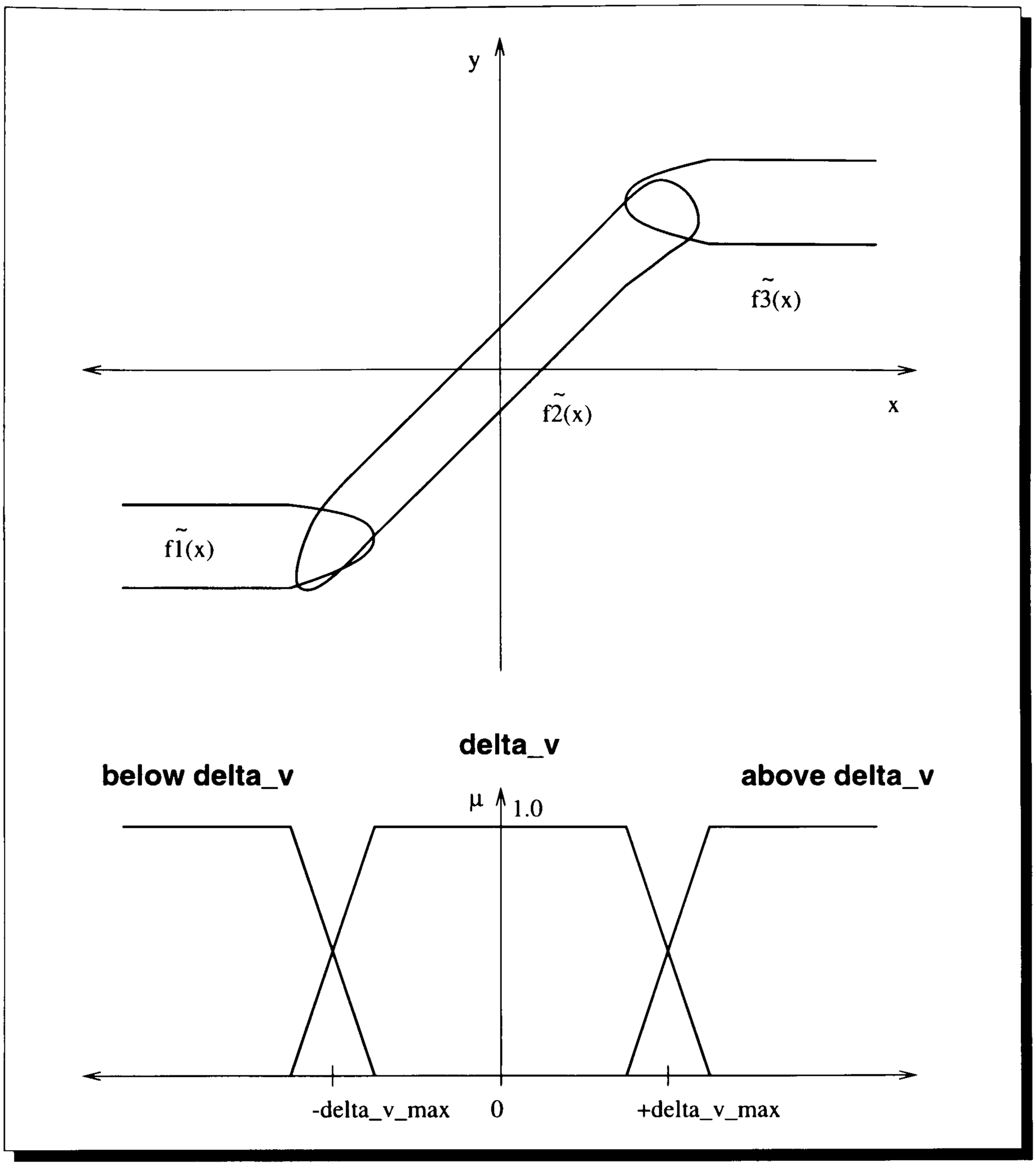


Figure 7.15: Fuzzy Relation Memory Modelling Nonlinear Systems

7.5 Conclusion

For input models a mixture of modelling approaches is convenient. If deep knowledge of the system is available the equation based modelling approach is preferred.

Often there is a lack of knowledge about internal structure and functionality

which makes it necessary to use a rule-based approach. Engineers typically use functional-based models mixing rule-based and equation based models. The new introduced Fuzzy Relation Memories (see Chapter 4) is a specialization of the Fuzzy Associative Memories. Fuzzy Relation Memories is a fuzzy rule-based model which maps fuzzy sets to fuzzifying functions. Using FRMs, humans can model an imprecisely known nonlinear systems in a natural way. An example was shown to give an idea of the power of such modelling approaches.

Chapter 8

Qualitative Fuzzy Simulation

Simulation is considered as part of the process of modelling and forecasting the behaviour of a system. The model's parameters, the model's input data, and the model itself might be imprecise. This imprecision is handled by the qualitative fuzzy approach, synonymously called **qualitative fuzzy simulation**. This chapter is summarizing the qualitative fuzzy simulation approach of this dissertation.

8.1 Qualitative Fuzzy Simulation

The qualitative fuzzy simulation approach used in this work is very similar to the approach of Bonarini and Bontempi described in their paper: "A Qualitative Simulation Approach for Fuzzy Dynamical Models" [Bonarini and Bontempi, 1994] as discussed in Chapter 2.3.

Therefore the qualitative fuzzy approach of this thesis is compared with Bonarini's and Bontempi's qualitative simulation approach.

Similarities of the Two Approaches:

- **Using Ordinary Differential Equations and Fuzzy Rules:** There is the possibility to describe models by ordinary differential equations and it is also possible to use fuzzy rules.

- **No Qualitative Models:** Qualitative models are neither used in the approach of Bonarini and Bontempi nor in the approach of this thesis. Because,

  1. in many engineering applications, a mathematical model could be available but it may be impossible to identify precisely its parameters,
  2. the engineers are not familiar with specific qualitative constraints like, e.g. M^+ or M^-, and
  3. it is always possible to build a mathematical model with imprecise defined parameters which has a similar behaviour as a qualitative model. A qualitative model is very often a less constraint model of a physical system then a mathematical model. Because qualitative constraints are defined where mathematical constraints could be used. This increases the imprecision of the simulation result using qualitative models. With the familiar mathematical model the engineer can use the parameters of the model to define these parameters so imprecisely that the simulation results is comparable to the results of the qualitative model.

- **Use of α-cuts:** Both approaches are based on the fuzzy logic approach that represent their fuzzy sets by α-cut sets. The representation of values this approach takes can be read in more detail in Chapter 3.

Differences of the Two Approaches

- **Imprecise Signals:** It is possible to describe dynamic knowledge, that changes during the simulation, or imprecise signals (Chapter 4) by **F**uzzy **R**elational **M**emories (FRMs Chapter 4.4).

- **Imprecise Modelling:** With **F**uzzy **R**elational **M**emories the possibilities of kinds of modelling is extended discussed in detail in Chapter 7.

- **Combining Imprecise Models:** Imprecise models represented by Fuzzy Relational Memories can be combined by arithmetic operations to a hierarchical

structure (see Chapter 5).

- **Interactive Evolutionary Algorithmic Approach:** The major differences of qualitative fuzzy simulation to the approach of Bonarini and Bontempi [Bonarini and Bontempi, 1994] is the method of solving imprecise differential equations by the "Interactive Evolutionary Algorithmic Approach" (6.6). Imprecise differential equations are ordinary differential equations whose initial values or equation parameters are not known exactly.

8.1.1 Summarizing the Features of Qualitative Fuzzy Simulation

The features of qualitative fuzzy simulation of this thesis can be summarized as follows:

- Ordinary imprecise differential equation-based models and rule-based models are accepted.
- The solution of differential equations using the "Interactive Evolutionary Algorithmic Approach" represents the solution space of all possible simulation results.
- If the model is precisely described in mathematical terms, the simulation produces the same results as traditional numeric simulators.
- Parameters of the models can be approximately known.
- Imprecise signals can easily be represented by Fuzzy Relational Memories.
- With Fuzzy Relational Memories imprecise models, also called fuzzy systems, can be built.
- Arithmetic operators can be used to build hierarchical structures of Fuzzy Relational Memories.

- Pure qualitative simulation is possible achieved at the symbolic level of the fuzzy linguistic variables.

8.2 Conclusion

The new reasoning method, "Approximate Modeled-Based Reasoning", has been introduced, that draws inference by simulating a model; a structured description of a system. The basic parts of such a model can be rule-based or mathematically modeled.

Both qualitative and fuzzy simulation can be achieved by qualitative fuzzy simulation. Most significant of the approach are first the **Fuzzy Relation Memories** which are a new type of value representation and model representation and second the **Interactive Evolutionary Algorithmic Approach** for solving differential equations whose parameters or initial conditions are not known precisely.

Part II

Qualitative and Fuzzy Analogue Circuit Design

PartII of the dissertation presents an application using the findings made in **Part I**. As discussed in Chapter 9 the engineering task of analogue circuit design is an excellent example for qualitative fuzzy simulation. Imprecision and vagueness is involved in the specification, modelling, and simulation of analogue system design.

PartII starts with a description of a **High-Level Framework for Imprecise Analogue Circuit Design (IACD)**. This new introduced high-level framework

- can handle imprecise specifications given by a circuit designer,
- supports the designer building a circuit paper prototype using qualitative fuzzy simulation, and
- is able to retrieve circuit cells from a pre-simulated circuit cell database by qualitative and fuzzy configuration design.

The problem of defining imprecise specifications is investigated in Chapter 10 separately.

The three main phases of the High-Level Framework for Imprecise Analogue Circuit Design

1. Circuit Paper Prototype Design Phase (Chapter 11)
2. Circuit Cell Characterization Design Phase (Chapter 12)
3. Ordering of Characterized Circuit Cell Design Phase (Chapter 12)

are described in detail.

Chapter 9

High-Level Framework for Imprecise Analogue Circuit Design (IACD)

This chapter starts with engineering design of nonlinear systems not known precisely. It discusses the engineering analogue circuit design which is of special interest. It describes the new **High-Level Framework of Imprecise Analogue Circuit Design** which is subdivided in a circuit paper prototype phase, a qualitative configuration design phase, and a fuzzy configuration design phase using a pre-simulated circuit database.

9.1 Engineering Design of Nonlinear Systems Not Known Precisely

Design is a peculiarly human activity: the desire to recast the world to suit our purposes is a defining characteristic of being human. Design in mechanical engineering, electrical engineering or for example architecture is defined differently, but informally accepted as different engineering disciplines. Mechanical engineering is in general

concerned with operation mechanisms, electrical engineering deals mainly with flows of signals, and architecture is especially interested in static entities. Rosenman and Gero ([Rosenman and Gero, 1994]) defined design most generally as:

> "Design implies a conscious purposeful activity to arrive at a state that did not previously exist in order to (presumably) improve some unsatisfactory (as perceived) existing state of affairs."

In engineering disciplines, design is the process by which a set of specifications and a set of available components are used to create a description of an artifact that satisfies these specifications. Engineering design differs from other kinds of design because it applies engineering methods to generate descriptions of useful devices. It consists of two major phases.

The analysis phase elaborates on the design specifications and determines the behaviour of the current design.

The synthesis phase decides on the individual devices and indicates their possible interconnections.

In circuit design devices can be single components like transistor, resistor, etc. or complete functional circuits called sub-circuits. Analogue circuit synthesis (also called schematic synthesis) creates very often both a circuit topology (also called schematic or circuit diagram) the interconnection of devices, and the correct sizing and biasing of the devices in that topology to meet input performance specifications. Before the design of circuits is discussed a more general view of design is examined which tries to categorize design problems in general. Bachmann, Bernardi, Klauck, and Schmidt [Bachmann et al., 1993] divide design problems into two groups. One group of design problems which need standard routines to solve the problem and one group which can only be solved if something new is created. This thesis takes a similar approach, as stated in Sgouros ([Sgouros, 1993]). Engineering design problems belong to three categories, depending on the type of search space they explore:

- **Routine Design Problems** are those in which the devices of a design are known and there is a specific method for assembling these devices (e.g. adapting DA-Converter, existing as a circuit cell in a library).

- **Innovative Design Problems** are those in which the devices of design are known, but there is no straightforward method for assembling these devices in a way that satisfies the design specifications (e.g. designing a new traffic light control system).

- **Creative Design Problems** are those in which not even the devices of a design are completely known[1]. Creative design also states that new systems have been designed.

Innovative design knows its devices exactly whereby creative design has only a vague idea about the devices which will build the new system. Williams [Williams, 1995] states:

> "Analogue design is more of an intuitive process than a hard science."

Vaguely known devices are typical for designing analogue circuits. There are several reasons why at an early stage of the design process the devices of the system are known not exactly:

- Slight variations of the component values might have big changes in the circuit behaviour (e.g. amplifier factor of a transistor). It is characteristic for an analogue circuit that a few components of a circuit might have complicated behaviours.

- One and the same analogue circuit might have different behaviours driven by different input (e.g. amplifier circuit is over-driven).

[1] Basically all design efforts of analogue circuits whereby no standard circuits cells are directly usable.

- The models during the design of analogue devices are rough approximations. First in real circuits the devices sometimes influence each other which is very difficult to consider. Second the time needed to simulate an analogue circuit which is modeled in detail would be too long.

The process of creative design of analogue circuits can be formulated to be more specific, based on the design definition of Neville ([Neville and Weld, 1994]).

Definition 9.64 (*Creative Design*):

Creative Design is defined as a process whose input is stated as:

- A set of possible imprecisely-described functional blocks. These functional blocks are described in terms of terminals, variables and precise or imprecise equations (see Chapter 7.2 *Imprecise Equation-Based Modelling*) or a set of rules relating the variables (see Chapter 7.4 *Imprecise Rule-Based Modelling*).
- Possible constraints on the number and type of legal connections between terminals.
- A description of an existing, incomplete device specified as a component-connection graph which makes hierarchical structures possible.
- A set of vague equations that denote the desired specifications of the complete design.

and which should result in:

- An analogue device connection graph which subsumes the existing device and whose equations are consistent and imply the desired behaviour.

Creative design rather stands at the beginning of a design process. At this stage the design engineer has a rough idea about the circuit topology and assumes a set of vaguely-defined circuit devices which might meet the specifications. The *circuit*

paper prototype design phase (Section 9.2) which is part of the analogue circuit design framework (Section 9.1) supports the designer by the creative design phase.

After a model has been found which consists of imprecise-defined circuit modules innovative design is used to find a realistic circuit. The framework of analogue circuit design uses *qualitative configuration design* and *fuzzy configuration design* (Section 9.4.3 and 9.4.4) to extract real circuit cells from a database. This automated phase of the design can be categorized as the innovative design phase of a design.

Before a framework can be developed the design process of analogue circuits of previous approaches will be analyzed. Four design phases are extracted and discussed in detail from the designer point of view (see Section 9.3).

9.2 Historical Survey of Analogue Circuit Design Tools

In this section only an overview is given. A detailed survey about existing design tools can be found in the Appendix B: "Historical Survey of Analogue Circuit Design Tools".

Design tools can be classified into bottom-up and top-down approaches. The bottom-up approaches are mainly used in the design of semi-custom integrated circuits whose library cells are based on a specific integrated circuit technology. Existing systems are:

- VITTOLD [Degrauwe et al., 1989]: layout generation of biquad and leapfrog SC filters,
- FPAD [Fares and Bozena, 1995]: design of operational amplifier circuit cells,
- system that design filters by genetic algorithms ([Horrocks and Khalifa, 1994], [Horrocks and Khalifa, 1995], [Horrocks and Arslan, 1995], and [Koza et al., 1996]).

Top-down approaches try to break down a high level specification to different level of descriptions until the integrated mask structure is reached. Top-down approaches are categorized into:

- *Algorithmic-Based Systems* try to apply algorithms to map the specifications to a real circuit. Examples are: operational amplifier compiler by Onodera, Kanbara, and Tamaru [Onodera et al., 1990] and component optimizer of a fixed circuit topology by DELIGHT.SPICE [Nye et al., 1988].

- *Hardware-Description Languages* allow to describe a system behaviourally using a hardware-language comparable with a programming language. At this level of abstraction the system can be simulated and if the system description meets the specifications it can be synthesized. Systems that need to be described and simulated continuously, like analogue circuits or mixed analogue and digital circuits, can use, for example, VHDL-A [VHDL-A Standard 99, 1999]. Synthesizing circuits described in the digital hardware-description language, VHDL [VHDL Standard 93, 1994], is already very successful. The synthesis of analogue circuits is still in its infancy. Especially if analogue systems are described behaviourally without any relations to existing analogue circuit cells.

- *Knowledge-Based Systems* use expert knowledge to design circuits. This artificial intelligence approach is known as expert systems. Knowledge is represented as facts, design rules, heuristics, algorithms, actions, and constraints. Given the specifications an expert system partially automates the design of circuits. Novice designers using these expert systems should be able to perform standard design tasks. Interesting analogue design expert systems are:

  - IDAC [Degrauwe et al., 1987] is a commercial interactive synthesis tool, that has different synthesis strategies for guiding the design of various amplifiers, oscillators, and filters. Each individual circuit synthesis can

be seen as an individual program.

- BLADES [El-Turky and Perry, 1989] uses expert heuristics, stated as rules, to determine an operational amplifier circuit topology given a circuit specification.
- OASYS's [Harjani et al., 1989] key concept is based on the assumption that the problem of transferring high level description of a circuit to an analogue circuit structure can be solved by decomposition of a hierarchical representation of circuits interactively.
- OPASYN [Koh et al., 1990]: synthesizes a layout of an optimized CMOS operational amplifier taking as input system level specifications, fabrication-dependent technology parameters, and geometric layout rules.
- STAIC [Harvey et al., 1992] is an interactive synthesis tool that designs CMOS and BiCMOS analogue integrated circuits by accepting structural and performance specifications as input by the user.

At present there is no analogue synthesis tool which is able to design a circuit not knowing the principle underlying circuit structure. The tools, for example, do not know when to design an operational amplifier or a filter just from the specifications given by the user. Little research has been done into design at a very high prototype level, e.g. block diagram design, taking into account that the specifications and the models defined by the designer are not known exactly. This imprecision can not be neglected during the design process when design mistakes should be avoided at an early design stage. This thesis tackles especially the problem of analogue circuit design at a very early stage of the design process.

9.3 Engineering Analogue Circuit Design in 4 Phases

Human-designed electrical circuits are replete with instances of the hierarchical reuse of previously designed sub-circuits. Indeed, design of a large electronic circuit would be impractical if the human designer had to start from first principles (e.g. Kirchhoff's current law and Kirchhoff's voltage law) and re-think the design of each sub-circuit each time it is needed. Therefore designers use a library of standard cells and customize them for the particular specifications.

Problems arise when there are no standard cells available or their existence in the database is unknown. Then the design engineer starts the design by sketching a prototype of an analogue circuit on a piece of paper using basic functional blocks. This network of functional blocks drawn on paper is called *circuit paper prototype* defined in Section 9.3.1.

The following sections describe the design of analogue circuits that have been partitioned into four phases.

9.3.1 Design Phase I: The Circuit Paper Prototype Design

Usually the design engineer starts the design by sketching a prototype of an analogue circuit on a piece of paper. This first prototype model of the circuit design consists of a set of functional blocks (e.g. voltage divider, amplifier, filter, etc.), which connected, should meet the given specifications most likely.

The designer is modelling the behaviour at the block level and describes the basic functional blocks as exact as known or needed at this stage of the design process.

Definition 9.65 (*Basic Functional Block*): Basic functional blocks are equation-based (Section 7.2), or functional-based (Section 7.3) high-level descriptions of a circuit behaviour represented as black boxes.

The description of the behaviour of analogue functional blocks at this level of abstraction is called block-level description. If a functional block is modelled entirely by a set of rules a basic functional block can also be called black box description.

Definition 9.66 (*Black Box Description*): A black box describes the behaviour of a functional block by a set of rules (Rule-Based Modelling; Section 7.4) stating the relations between the input data and output data.

The expert very often has knowledge about the setup of a functional block. The expert is able to describe a functional block by a set of basic mathematical relations (constraints), or functional sub-blocks connected and build up hierarchically. The internal structure of these functional blocks are known by the expert. Design experts use deep knowledge to represent the behaviour of such a functional block.

But sometimes it is impossible to build a model from basic modelling components of the real physical world. Only shallow knowledge is available to describe the functional block.

If there are both descriptions available for one particular functional block the inconsistency between both must be resolved. Gyooseok and Fishwick [Gyooseok and Fishwick, 1997] developed a method to resolve the inconsistency between shallow modelling and deep modelling.

All the modelling approaches have in common that they do not necessarily have something to do with the internal circuit schematic of the black box. The internal details of the circuit topology is hidden in the black box. This gives the designer the possibility to model different types of amplifier with the same model. It does not depend on a specific amplifier architecture.

The network of functional blocks drawn on paper at this stage of the design process is the first attempt and is by no means correct. The complete network consists of vaguely known functional blocks which do not even necessarily represent an analogue circuit.

The next step in the design process is to check whether the behaviour of the draft

network is correct by behavioural simulation considering the vagueness involved in the network. Behavioural simulation and modelling is useful because they help designers reduce design time. Verifications of the behaviour of the system can be investigated before detailed circuit implementations have to be evaluated which is very time consuming. The behavioural simulation at this vague level of abstraction is often done on paper directly, propagating the specified signals through all functional blocks until the complete block network is simulated using rules of thumb. Electrical conditions and behavioural information about the circuit are added by labeling the terminals of the network drawn on paper with signal sketches (also called graphs, or diagrams, or fuzzy curves introduced in Section 4.4).

Graphs are very important add-ons to the circuit. They explain and visualize the dynamic behaviour of the circuit. As Larkin and Simon ([Larkin and Simon, 1987]) state: "A diagram is (sometimes) worth ten thousand words." Signal sketches convey information in a compact form and illustrate the behaviour of a system. They make the system verifiable by other designers at an early stage of the design process.

After Design Phase I of the design process the design engineer produces a circuit paper prototype.

Definition 9.67 (*Circuit Paper Prototype*): A circuit paper prototype is a circuit network built of basic functional blocks. Their terminals are labeled with behavioural signal sketches for visualization of behavioural information.

An essential feature for the circuit paper prototype design is the behavioural simulation of analogue circuits, especially reasoning with diagrams, also called graphs. The following features are of interest:

- The models of the analogue blocks have to be *general.* They must be independent from the internal circuit topology. For example, different types of Flip-Flops can be modeled with the same behavioural model.

- It must be possible to add *second-order effects* to the behavioural models of the analogue blocks. They are occasionally important for the designer to get a realistic idea of the overall performance of the system.

- For analogue circuits, *time* and *frequency* simulations are important domains. The simulation has to be performed in whatever domain is of interest to the designer or whatever domain is necessary to obtain the simulation results.

The circuit paper prototype design enables the design engineer to verify the requirements of the circuit at a very high-level of abstraction and avoids early design faults.

9.3.2 Design Phase II/III: Acquisition of Real Analogue Circuit Cells

Having an analogue circuit model of basic functional blocks which meets the specified requirements of a real circuit. Each realizable block must be replaced by a real circuit cell.

Definition 9.68 (*Realizable Block*): One or several basic functional blocks build a realizable block which in future is replaced by a real analogue circuit.

Whether one or several functional blocks build a realizable block depends on the level of detail the designer used to model the circuit paper prototype. An amplifier circuit could be modeled by a simple multiplication function or more realisticly by several basic functional blocks which consider attributes like input losses, power dissipation, etc.

Suppose a circuit designer is looking for a real circuit cell which replaces the complete developed model from the circuit paper prototype design phase. Because of the huge library of analogue circuit cells, a design engineer proceeds this design phase in two steps:

1. **Circuit Cell Characterization:** Circuit cells of the database are chosen which have the same qualitative behaviour as the realizable model developed at the circuit paper prototype design. Analogue circuits which have equal qualitative behaviour are grouped to circuit cell families. A circuit family, for example, is a family of amplifier circuits, current mirror circuits, AD-converter circuits, etc.

   Definition 9.69 (*Circuit Cell Characterization*): Circuit cell characterization is looking for circuit cells which have the same qualitative behaviour as the realizable block developed at the circuit paper prototype phase.

2. **Ordering of Characterized Circuit Cells:** Circuit cells of a circuit family are chosen which will meet the requirements most likely. An analogue circuit cell database, for example, consists of more than one kind of amplifier circuit. There are high-frequency amplifier circuits, high-impedance amplifier circuits, etc. An exact matching of an analogue cell in the library and a realizable block is unrealistic since too many parameters of the cell can be changed for customization and vagueness of the circuit paper prototype model.

   Definition 9.70 (*Ordering of Characterized Circuit Cells*): Ordering of characterized circuit cells is finding a circuit from a set of circuits, the circuit family, which meets the requirements best.

Both problems can be seen as configuration-design problems. One of the most well-studied design problems is configuration design. Configuration-design is stated in Wielinga and Guus ([Wielinga and Guus, 1997]) as:

> "Take a set of predefined components, add the search for an assembly of components that satisfies a set of requirements and obeys a set of constraints, and you have configuration-design."

Looking for circuit cells which have the same qualitative behaviour as the realizable block at the developed circuit paper prototype phase is done at the qualitative level

of configuration-design, called *qualitative configuration-design.*

Definition 9.71 (*Qualitative Configuration-Design*): Qualitative configuration-design takes a set of circuit cells from the database, adds the search for an assembly of these cells that satisfies a set of possible constraints and equations by "qualitative simulation" [2].

Finding a circuit from a set of circuits, the circuit family, which meets the requirements best is related to creative design. Creative design, as stated in Section 9.1, in analogue circuit design is finding the real circuits from a database which meet the requirements best for each realizable block, taking into account that at this design stage the requirements are neither well defined nor known precisely. This uncertainty is modeled using the fuzzy approach which results into *fuzzy configuration-design.* Fuzzy configuration-design of analogue circuits can be defined as follows:

Definition 9.72 (*Fuzzy Configuration-Design*): Fuzzy Configuration-Design takes a set of imprecisely-described functional blocks which have been extracted from a database of analogue cells, adds the search for an assembly of these blocks that satisfies a set of possible constraints and vague defined equations by "fuzzy simulation" [3].

Summarizing Design Phase II/III:

Design Problem	**Kind of Configuration-Design Used**
circuit cell characterization	qualitative configuration-design
ordering of characterized circuit cells	fuzzy configuration-design

The qualitative configuration-design, as introduced, gives the designer a categorization of the circuits from a circuit cell database without any valuation of the

[2]Qualitative simulation is a sub-set of "Qualitative Fuzzy Simulation" (see Part I). The qualitative values are mapped to fuzzy values and after a qualitative fuzzy simulation remapped to qualitative values.

[3]Fuzzy simulation is a sub-set of "Qualitative Fuzzy Simulation" (see Part I).

found circuit cells. Further it is computationally less expensive then the fuzzy configuration-design stage.

9.3.3 Design Phase IV: Sizing and Biasing of Analogue Circuits

Having found a circuit cell, which is most likely to meet the requirements best, the components of the circuit have to be sized. Sizing is finding the optimal solution in terms of geometric area dimensions of the components on the circuit carrier (the waver), total power needed for the circuit, overall performance of the circuit, and other attributes stated in an objective function to be optimized. An optimization-based design procedure usually performs the sizing. The evolving circuit is perturbed slightly by the optimizer, and the simulator is invoked to evaluate the resulting performance, which is then used in a cost function that measures, for example, deviation from the desired specifications. This works quite well, if the circuit to tune is already close to a good design, but cannot typically be used on a circuit that is completely unsized and unbiased.

9.3.4 Design Reasoning — Design Phases Related to Reasoning

How are the different design phases related to reasoning?

To say that reasoning is the "manipulation of available knowledge" is very much of a simplification. There are many kinds of complex reasoning schemes that can be applied to available knowledge bases, e.g. case-based reasoning, rule-based reasoning, model-based reasoning, fuzzy reasoning, qualitative reasoning, etc.

As stated in Kleiber and Kulpa [Kleiber and Kulpa, 1995] any single reasoning approach is not sufficient to cover all important cases and problems of analysis of physical systems. The future of computer-assisted analysis of physical systems

belongs to hybrid systems.

Design reasoning about physical systems is a hybrid reasoning approach. It combines several more specific kind of reasoning tasks, in particular the following:

- **Model-Based Reasoning:** Model-based reasoning tries to process information (e.g. new behaviour) about a physical system using models about the systems. Models are created by the designer during the circuit paper prototype design phase.

- **Case-Based Reasoning:** The designer uses or adapts previous experience to meet the demands of a newer situation. Analogue circuit designers use previous designed circuits to meet the demand of a new specification. Both qualitative configuration-design and fuzzy configuration-design belong to case-based reasoning.

- **Qualitative Reasoning and Fuzzy Reasoning:** Imprecision of the systems have to be taken into account by using the qualitative or the fuzzy approach. Both reasonings are combined by qualitative fuzzy simulation (Section 8.1).

The designer must build a model to describe the system and uses simulation to predict the possible behaviour of the system. The model building task has to be done by the designer at a very high level of abstraction, reusing previous models. At the circuit paper prototype design phase model-based reasoning is used as building models of the system by the designer and simulate the system. The goal of the simulation in the circuit paper prototype design is to find out whether the model justifies the given required specifications or not.

Qualitative configuration-design and fuzzy configuration-design (discussed in detail in Chapter 12) is part of the case-based reasoning approach. Case-based reasoning is a more general approach adapting previous experience to meet the demand of a newer situation while configuration-design is searching for suitable circuit cells in a database adapting it to a new description of a basic functional block.

Central to this thesis is simulation and modelling of imprecise physical systems. The qualitative and the fuzzy reasoning offer the modelling and simulation of imprecise systems. A combination of both, called qualitative fuzzy simulation (see Section 8.1) is used to do model-based approximate reasoning.

9.4 High-Level Framework of Imprecise Analogue Circuit Design (IACD)

Discussing the process of designing analogue circuits by the design engineer in the previous Section 9.3, a new high-level framework for the design of imprecise modeled and specified analogue systems has been developed based on qualitative fuzzy simulation approach.

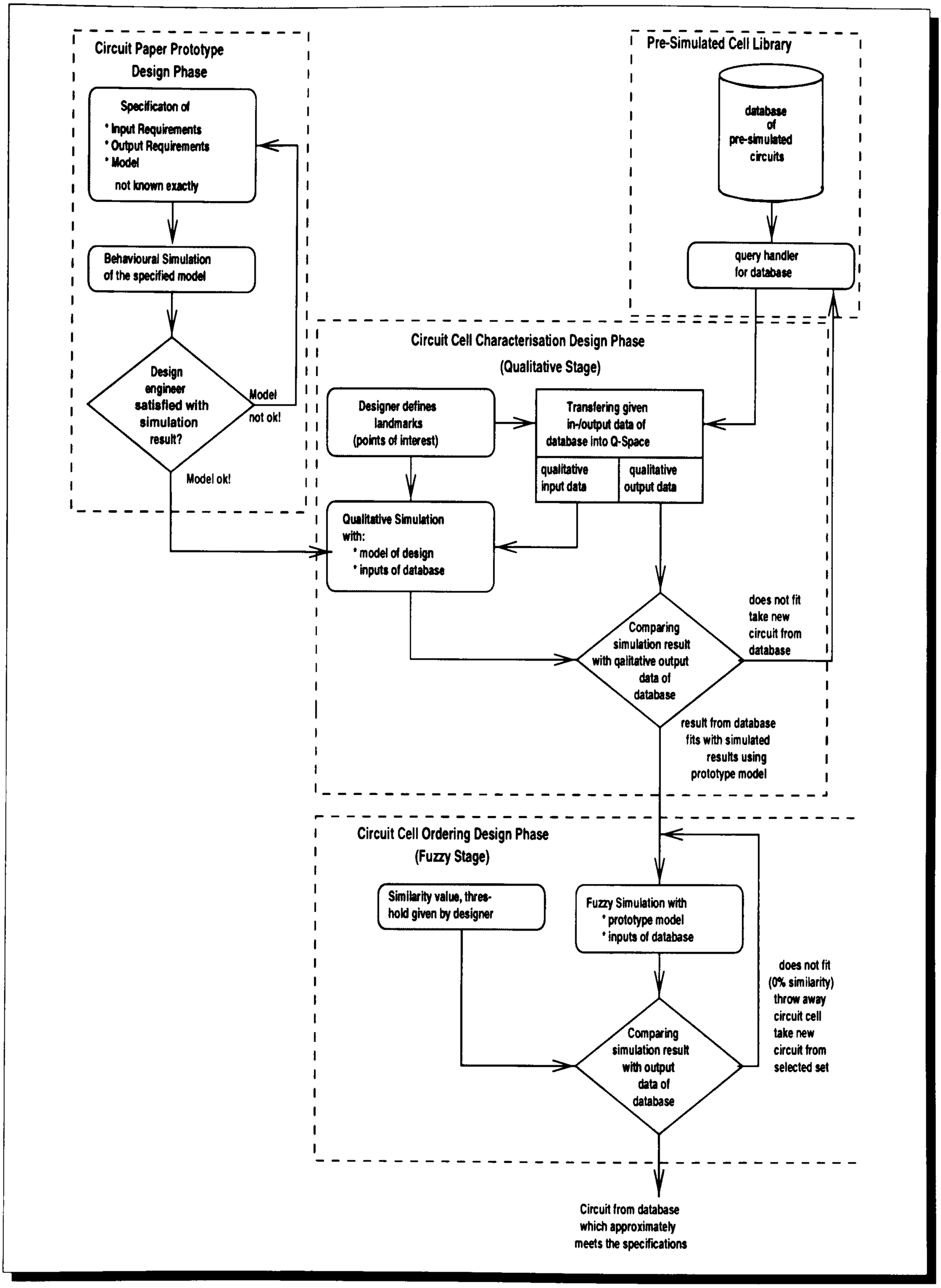


Figure 9.1: Framework for Design of Analogue Circuits

The framework (Fig. 9.1) of the design of analogue circuits covers three important design phases of the four previously discussed:

1. **Design Phase I: Circuit Paper Prototype Modelling Phase:** Support of the construction of the circuit paper prototype model. Verification of the model prototype by behavioural simulation, to check if requirements are met (see Section 9.4.1).

2. **Design Phase II: Circuit Cell Characterization Design Phase:** Characterizing the analogue circuit cells of a given library (database). The cells which have the same qualitative behaviour are selected (see Section 9.4.3).

3. **Design Phase III: Circuit Cell Ordering Design Phase:** Ordering the previously found analogue circuit cells according to their fitness meeting the specifications (see Section 9.4.4).

4. **Design Phase IV: Sizing Design Phase:** The sizing and biasing of the analogue circuit cells is not covered by the framework. There are several tools available, e.g. DELIGHT.SPICE [Nye et al., 1988], ECSTACY [Shyu and Sangiovanni-Vincentelli, 1988], and ADOPT [Lai et al., 1988], for finding the optimal solution in terms of area, power, and overall performance and optimizing an objective function.

In the following Sections each issue of the framework (dash-line box) is explained in detail.

9.4.1 Design Phase I: Circuit Paper Prototype Design Phase

Before a circuit can be designed, the requirements, usually given by the customer, have to be specified for the design process. These specifications consist of input data and output data often not known exactly. The input data states information that

is needed to simulate the circuit model. The output data states the requirements which have to be met after simulating the model. After the specification the designer builds a model from basic functional blocks which are also not known precisely. To know whether the requirements for the circuit are met or not by the model, the model is simulated using qualitative fuzzy simulation (Part I).

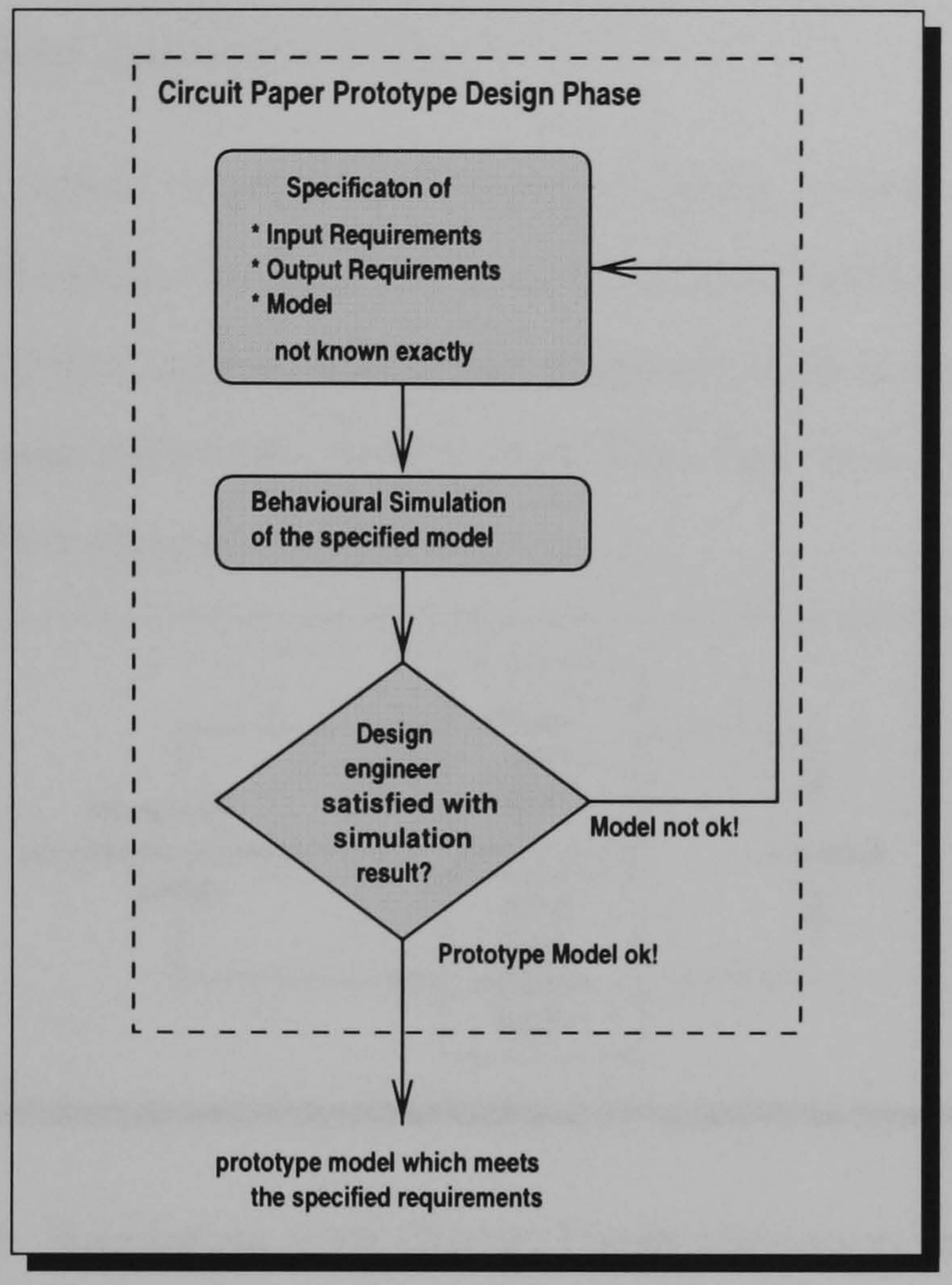


Figure 9.2: Circuit Paper Prototype Design Phase

In Fig. 9.2 it is stated that the input requirements, the output requirements, and the model have to be specified. Both the requirements and the model do not need to be known exactly. After a qualitative fuzzy simulation the designer gets simulated output data. The simulated output data and the specified output data is compared using a similarity measurement defined in Chapter 5. The design engineer can decide whether the model meets the specified requirements or not, using the similarity measurement. This enables the designer to verify whether the requirements of the

prototype model are met or not at quite an early design stage.

9.4.2 Pre-Simulated Circuit Cell Library

Why is a pre-simulated circuit cell library needed? There are two possibilities for checking whether a circuit cell of the library has the same behaviour as the circuit paper prototype model developed or not:

1. The specified inputs to simulate the circuit paper prototype model are taken and each circuit from the database is selected and simulated with these input. Then the simulated output results are compared with the output generated at the circuit paper prototype design phase (see Fig. 9.3). If they are suitable the circuit cell is chosen.

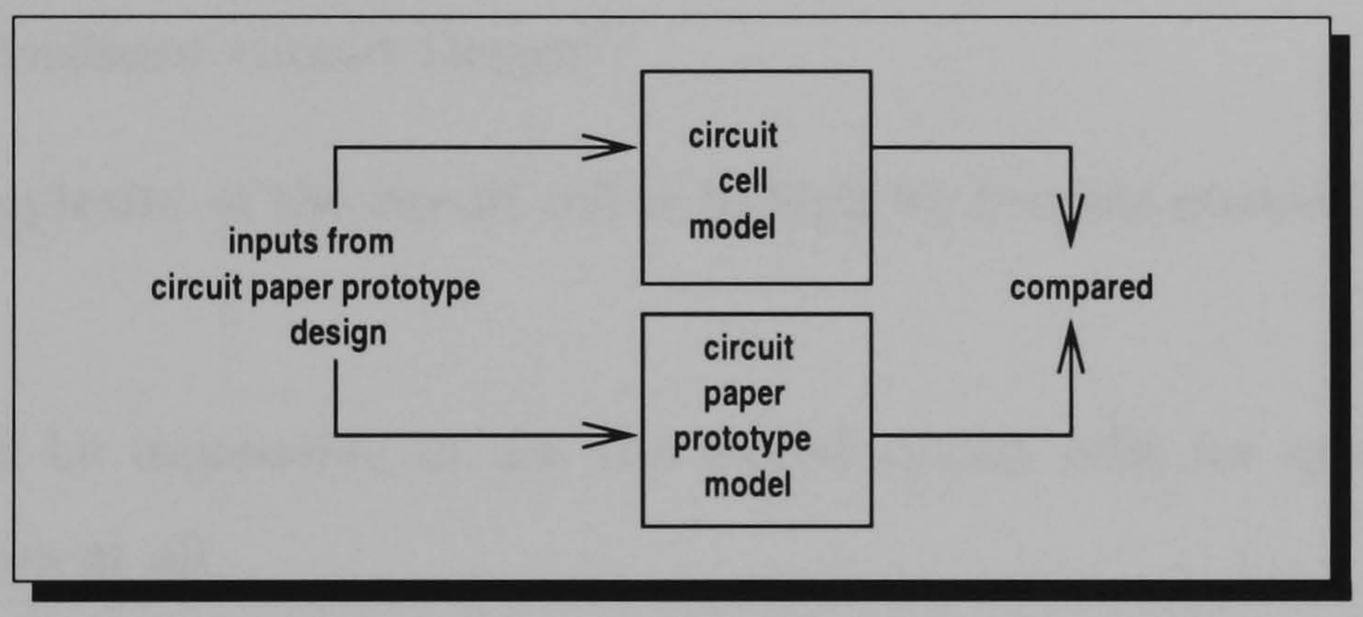


Figure 9.3: Simulation with Circuit Paper Prototype Input Data

2. When the circuit cell has been developed, the circuit cell was simulated to check whether it is correct or not. These simulations were detailed, typical, and specific for each particular circuit cell. Suppose these input/output data has been stored with the circuit cell in a library. The underlying assumption of this IACD approach is that the circuit paper prototype model and the circuit cell model have the same behaviour Therefore given the input data from the database would cause the same simulation result. The input data from the database is taken to simulate the circuit paper prototype model and the output

from this simulation is compared with the stored output from the database (see Fig. 9.4).

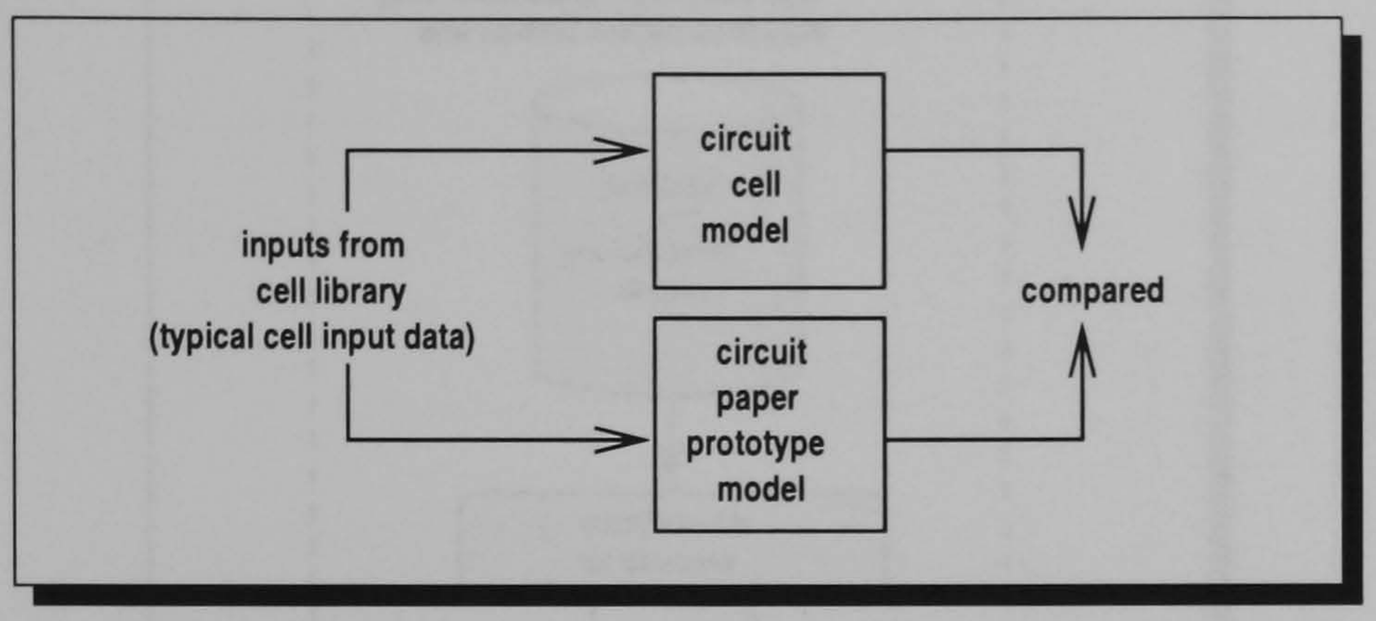


Figure 9.4: Simulation with Cell Library Typical Input Data

There are reasons for choosing the second possibility for the "High-Level Framework of Imprecise Analogue Circuit Design":

- The complexity of the circuit cell is to high for feasible qualitative fuzzy simulation.
- It might be impossible to use the stored circuit cells for qualitative fuzzy simulation at all
- The circuit cells have been designed for a particular purpose using different input data than the circuit cell was designed for (e.g. large signals rather than small input signals) could generate correct output data. The real circuit never shows such behaviour because the model of the circuit cell might be incorrect at such input signals.

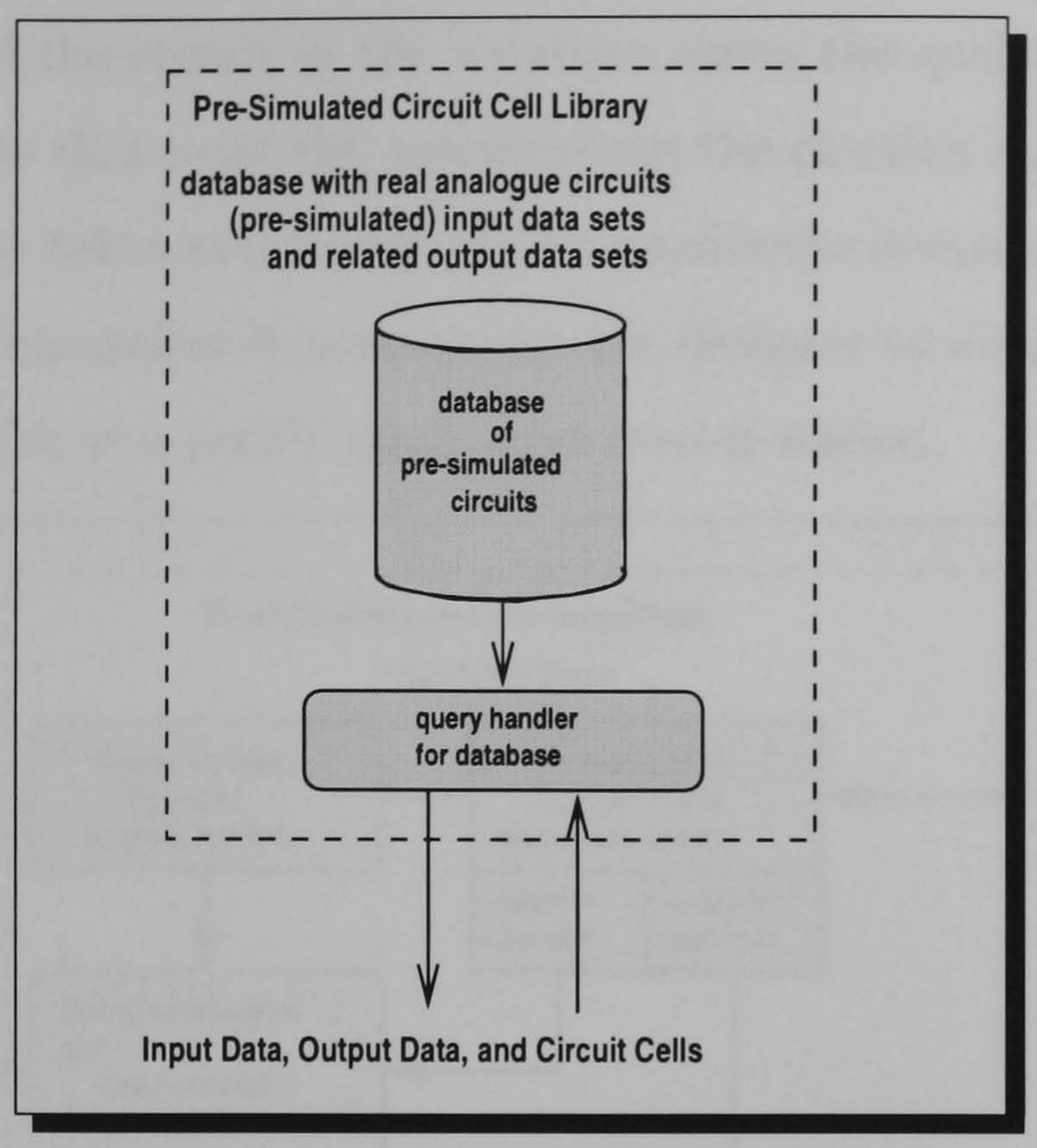


Figure 9.5: Circuit Cell Library

A pre-simulated library (Fig. 9.5) contains real analogue circuit cells including typical input/output data sets. An analogue circuit cell is pre-simulated when a set of input data is used to simulate a set of output data. The set of input data is typical data for testing this specific circuit cell. A rectifier circuit, for example, does have different input data than a sample-and-hold circuit. The input data does not have to be the set of input data specified in the circuit paper prototype design.

9.4.3 Design Phase II: Circuit Cell Characterization Design Phase

Based on the qualitative simulation approach the circuit cells of a library are characterized. Each circuit cell in the library has a set of real output values related to a set of input values. Using the circuit paper prototype model and the input the qualitative simulation determines the possible output behaviour of the model prototype. After comparing the prototype model's output with the output from the database

it can be decided if the circuit in the database meets the qualitative behaviour or not. For the circuits that meet the specifications the circuit's input and output set from the database is taken and transferred to qualitative domain pre-defined by the design engineer. This makes it possible for the designer to compare and interpret the simulation results at a purely qualitative level if wished.

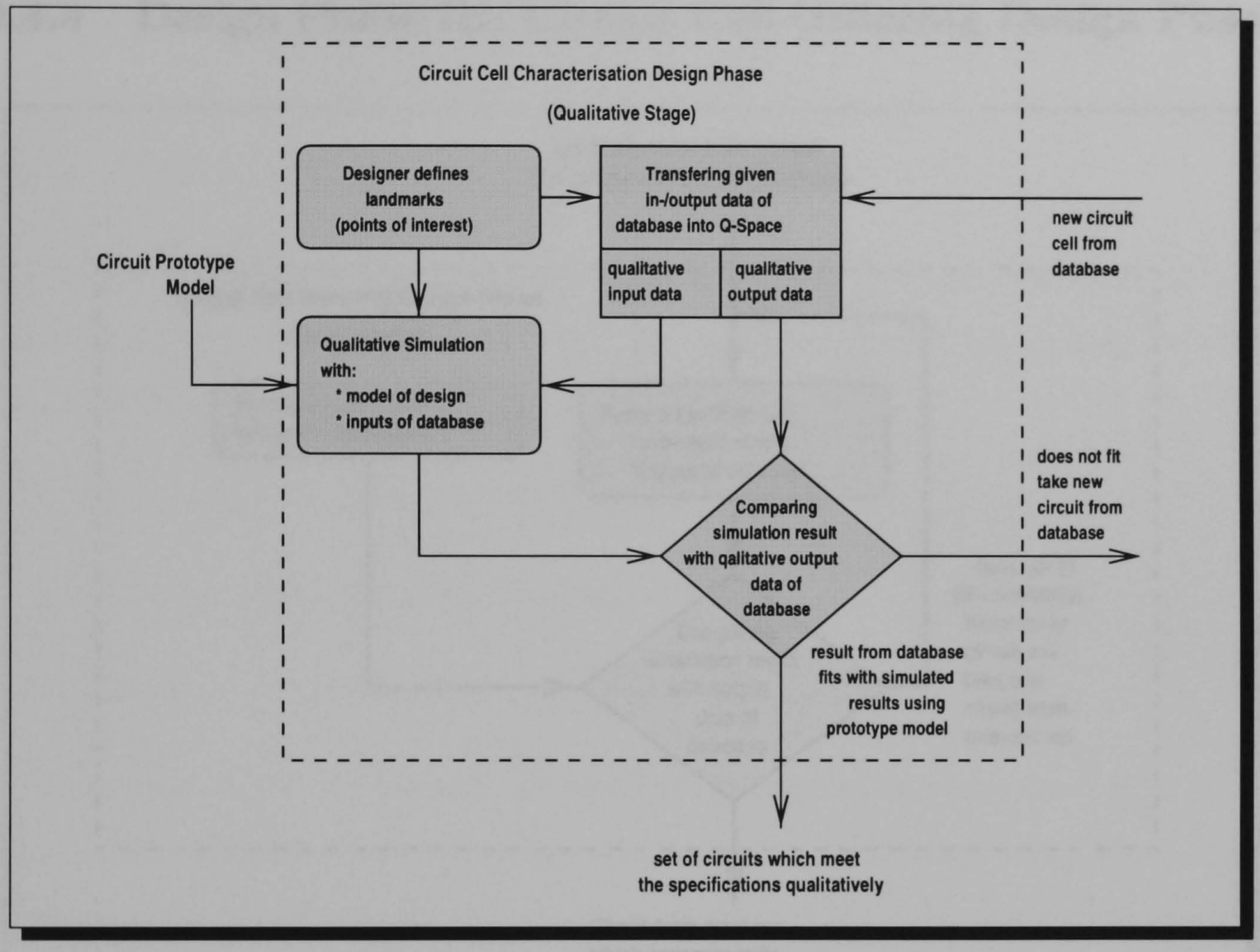


Figure 9.6: Circuit Cell Characterization Design Phase

Fig. 9.6 shows the qualitative stage of the qualitative configuration design. Using the model prototype and the input/output sets of the database leads to a characterization of the database. Only the circuits that have the same qualitative behaviour as the model prototype are selected. It is assumed that the model prototype is a principle representation of the found analogue circuit cell of the database. Qualitative simulation generates the complete behaviour of a model including behaviour which has no realistic justification. The qualitative configuration-design in this the-

sis is looking for an assembly of circuit cells which can replace a realizable block starting to test each circuit cell one by one. Satisfying a set of requirements and a set of constraints is comparing qualitative simulation results of the model prototype simulated with the input data given with the output data given by the library.

9.4.4 Design Phase III: Circuit Cell Ordering Design Phase

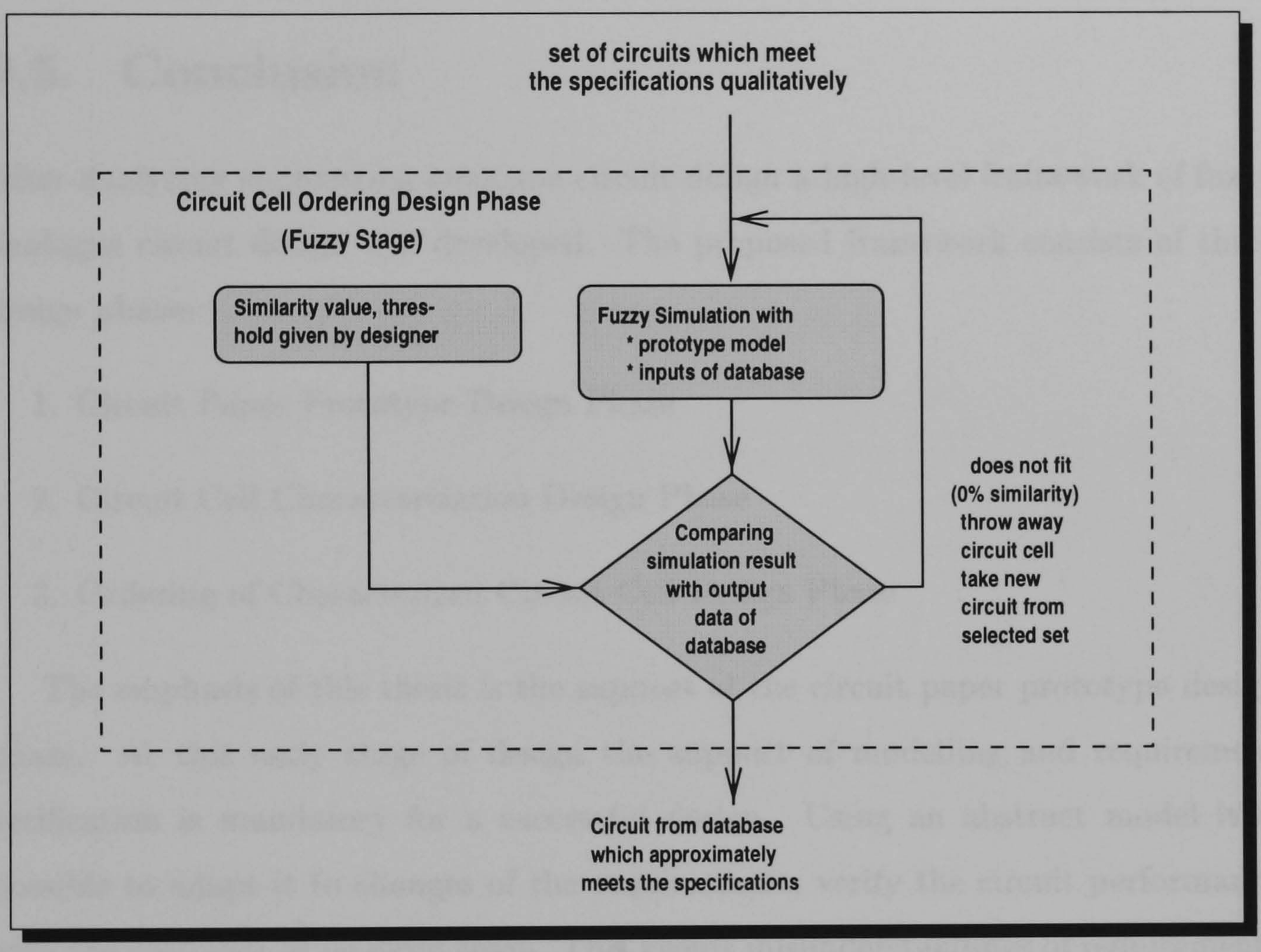


Figure 9.7: Circuit Cell Ordering Design Phase

Exact configuration-design in analogue circuit design is not feasible. It is unrealistic to find an analogue circuit cell which meets the specifications completely and exactly. Besides it is not necessary, since there are specifications which are mandatory and specifications which are optional. Designers often have to compromise. Fuzzy configuration design has only the circuit cells from the characterizing cell design

phase. The search for an assembly of devices looks for circuit cells one by one. The circuit cells are ordered according to the satisfaction of a set of requirements and a set of constraints. Which of the circuit cells, developed at the circuit paper prototype design phase, represents the model best, is determined by comparing simulation results of the circuit prototype using the input data and the output data given by the library of each circuit cell.

9.5 Conclusion

After analyzing engineering analogue circuit design a high-level framework of fuzzy analogue circuit design was developed. The proposed framework consists of three design phases:

1. Circuit Paper Prototype Design Phase

2. Circuit Cell Characterization Design Phase

3. Ordering of Characterized Circuit Cell Design Phase

The emphasis of this thesis is the support of the circuit paper prototype design phase. At this early stage of design the support of modelling and requirement verification is mandatory for a successful design. Using an abstract model it is possible to adapt it to changes of the requirements, verify the circuit performance with the customer at an early stage. This avoids misunderstandings of requirements between designer and customer and prevents time consuming redesigns at a later design stage. The developed model is very high-leveled and is not related to any real circuit model. The model can be developed independently from any real circuit model.

After the circuit paper prototype stage the search for real analogue circuit cells is automated by the newly introduced two design phases of

1. "Circuit Cell Characterization Design Phase" solved by qualitative configuration-design and

2. "Ordering of Characterized Circuit Cell Design Phase" solved by fuzzy configuration-design.

The sizing and biasing of the found circuit cells is not covered by this thesis.

Chapter 10

Defining Imprecise Specifications

Designing a dynamic physical nonlinear system can be expressed, as finding a system that fulfills a given specification. Typical representatives for physical nonlinear systems are analogue circuits. Beside the difficulty of finding a correct analogue circuit, there is the problem of circuit description. At first the circuit description consists of nothing but a set of wishes which are neither precise, detailed, nor even complete.

The main contents of the chapter is a detailed discussion about imprecise parameter specification for nonlinear systems. It outlines a possible user interface to support analogue circuit designer to specify imprecise input/output data.

10.1 Imprecise Design Activity

Questioning an experienced design engineer makes it clear that there is a lot of uncertainty involved during the design process. This imprecision can be summarized by two important aspects:

1. **Vague Specifications:** An important and relatively unaddressed aspect in design is the fact that the specifications of a system are not necessarily well defined. There are some specifications usually defined by the designer which

can be categorized into:

- **Must-Specifications** which must be met by the designed system.

  They are requirements given by the customer of the system. Must-Specifications do not have to be defined by a singleton, e.g. a single worst-case specification. The customer might define the requirements of the system portioning them into best-case and worst-case system requirements. Very often specifications are too constrained, ruling out designs the customer would accept at the end, because of the lack of more detailed levels of the must-specifications, like worst-case, acceptable-case, best-case, etc. Must-Specifications are defined in this work by fuzzy values representing two levels of specifications that are called permitted and not permitted sections.

- **Wanted/Optimal-Specifications** which should be met by the designed system.

  These specifications are wanted by the customer. They do not have to be met but the better these specifications are met the better the design. These specifications are defined by fuzzy values by sections that are called optimal sections.

- **Should-Specifications** that should be defined by the designer in order to simulate the system to be designed.

  During the design of a circuit very often there arise the necessity for additional definitions of parameters in order to simulate the system. These specifications are guessed by the designer. They are by no means well-known and the uncertainty of such a guess should be taken into account. Should-Specifications are represented by fuzzy values by sections that are called optimal, permitted, and not permitted sections.

2. **Vague Models:** During a design process there are different levels of details of the circuit models. The model of a circuit must be as detailed as is necessary. They all share the feature that the models are not modelling the real circuit exactly.

   The model is more or less vague at the start of the design process, when a system does not exist yet. A designer of such a model is able to state the imprecision and vagueness involved in the description as described in Chapter 7.

These two aspects for describing specifications and models not known exactly are founded by the qualitative fuzzy approach discussed in more detail in Part I "Qualitative Fuzzy Simulation".

10.2 Specification of Imprecise Nonlinear Systems and Design Requirements

To specify an analogue circuit there are three major areas of interest, which can be categorized as environment description, interface description, and behavioural description as similar to Mueller-Glaser [Mueller-Glaser and Bortolazzi, 1990].

Environment description are specifications concerned with e.g. temperature, humidity, mechanical stress, etc. the system is operating.

Interface description defines how the system is interacting with the environment.

Behavioural description of a circuit, the emphasis of this thesis, is most significant for finding a circuit which will function as intended. The behavioural description concerns the circuit itself and is characterized by its functions.

The behaviour of a system describes how it does something. Whereas the function of a system describes what specific task it performs. The behaviour can be defined

by input/output relationships (e.g. transfer functions) or indirectly by stating the input data and output data of the analogue circuit. Fig. 10.1 gives an overview of the possible entities covered in this work.

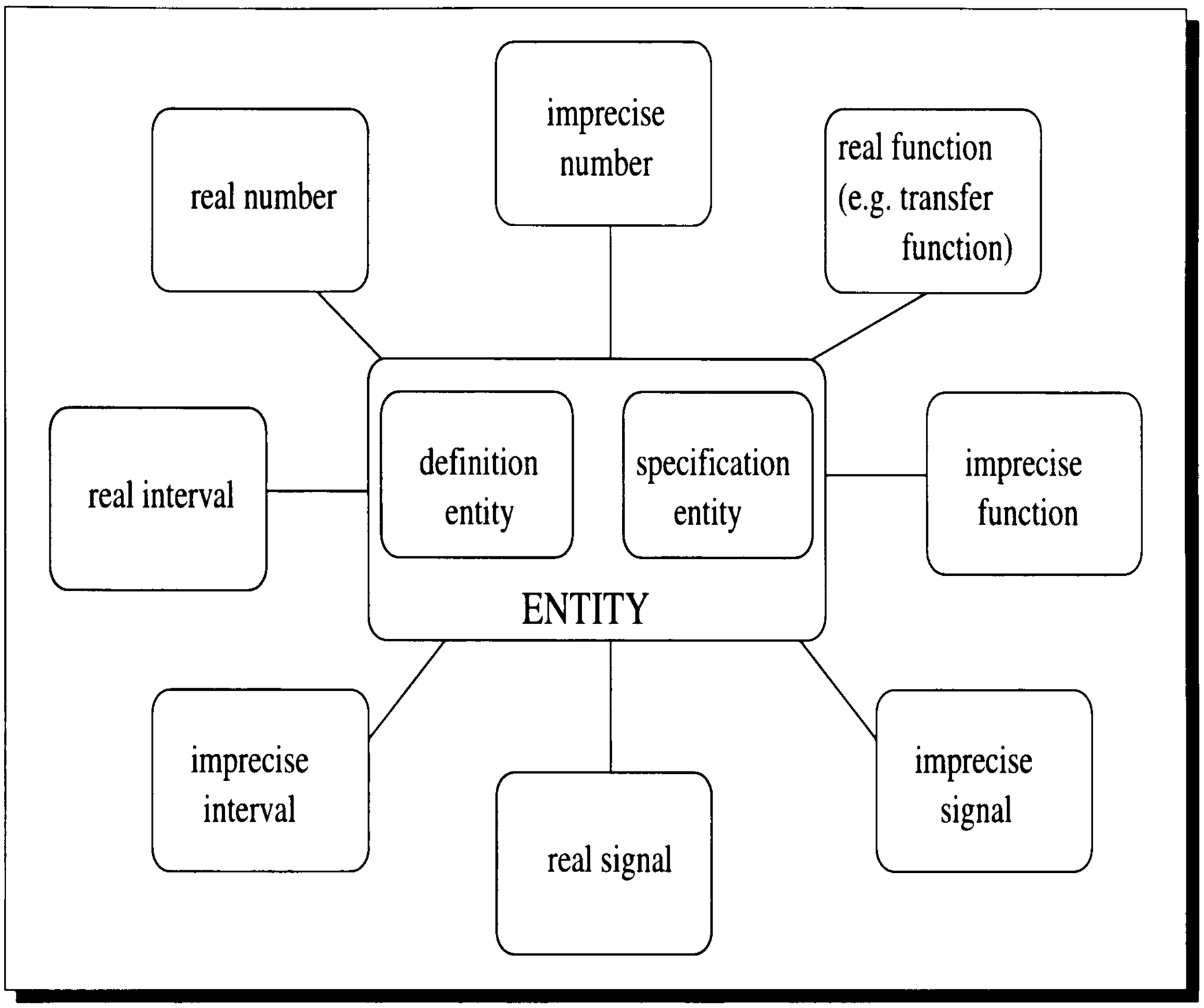


Figure 10.1: Entity Overview

The specifications (Fig. 10.1) can be categorized into numbers, intervals, functions, and curves, real and imprecise respectively. The distinction between **definition entities** and **specification entities** is striking.

Definition entities are entities which are input into the system not known exactly for example voltages or currents continuous (e.g. DC-5V, DC-1mA) or discontinuous (e.g. signals, frequencies, AC-5V).

Specification entities are used to express the *optimal*, *permitted*, and *not permit-*

ted output of the system. Specification entities are not input to the system; they are used to judge the output generated by a simulation.

The input data are descriptions of observed, or intuitive given data whereas the wanted output data are data descriptions which take into account that there are sections where the data is *permitted*, *not permitted*, or *optimal.*

The special interest of this dissertation is the kind of imprecise specification which is usually defined by signals (also called graphs) sketched on paper by the designer. Humans seem to prefer graphs, the visual representation of information. In text books, data-sheets and in training unexperienced designers the graph representation is frequently used.

10.3 Defining Input Data Not Known Exactly

There are two major kinds of input data definitions used by designers for specifications. Input data that is constant during simulation is modelled by fuzzy numbers. Data that changes depending on other data (e.g. time) of the system is modelled by fuzzifying functions or Fuzzy Relation Memories.

10.3.1 Definition of the Membership Function

Fuzzy sets are defined by their membership functions. The problem for a circuit designer is to specify the membership function of fuzzy values. How does a designer have to interpret the different *levels of presumption* (defined in Chapter 3). A circuit design engineer has not necessarily got knowledge about membership function definitions. There are several methods for assigning membership functions to fuzzy variables, e.g. intuition, inference, rank ordering, neural nets, genetic algorithms, meta rules, etc., summarized in the book: "Fuzzy Logic With Engineering Applications" by Ross [Ross, 1995]. Membership functions are defined in this thesis using intuition supported by a prototype user interface.

Definition 10.73 (*Intuitively-Defined Membership Functions*): The designer develops the membership functions through his/her own innate intelligence and understanding.

These membership functions are functions of context and highly depending on the person developing them. Especially when defining a linguistic variable like frequency in terms of e.g. {*low*, *medium*, *high*}. A low frequency in one system (e.g. satellite receiver) might be high in an other system (e.g. current supply).

Of advantage is that the person developing the membership functions can easily interpret the result of a simulation. Other designers might have difficulties or interpret results wrongly because they have a different understanding of the membership functions.

10.3.2 Fuzzy Numbers/Intervals Model Vague Constant Input Data

Vague constant data is modeled by regular fuzzy numbers or fuzzy intervals. Suppose the circuit designer wants to define a vague interval. The user interface should support the designer to state the vagueness involved in the data. Fig.10.2 is an example for an user interface for defining a fuzzy interval. The user interface shown in Fig.10.2 is part of a tool which was experimentally built during this work.

With the user interface shown in Fig.10.2 the designer defines the significant values of the membership function (Fig. 10.3), which are the lower and upper boundary of the most likely interval and the lower and upper boundary of the interval where the value is just in between. These values are defined by the designer's intuition.

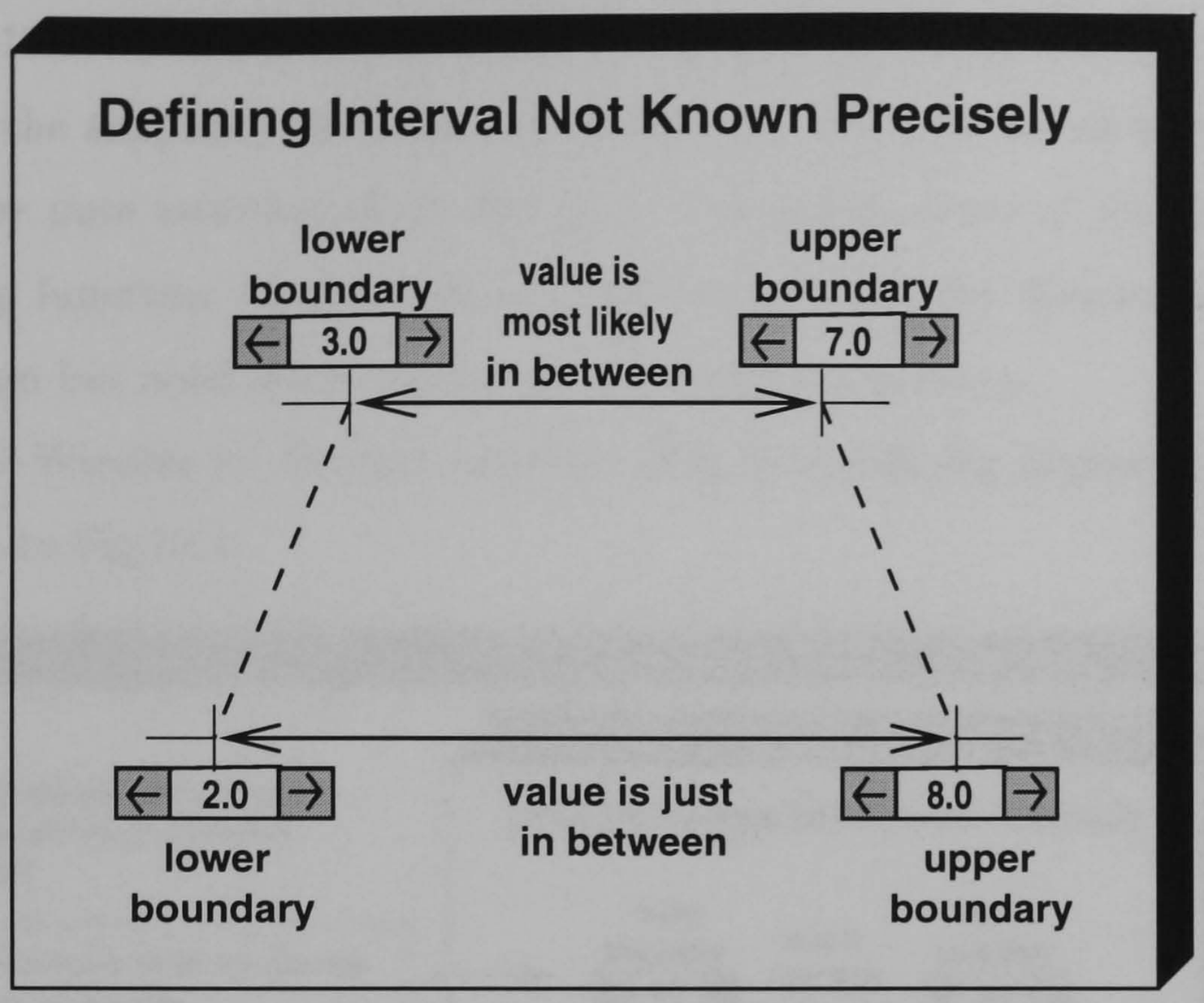


Figure 10.2: User Interface for Defining Vague Intervals

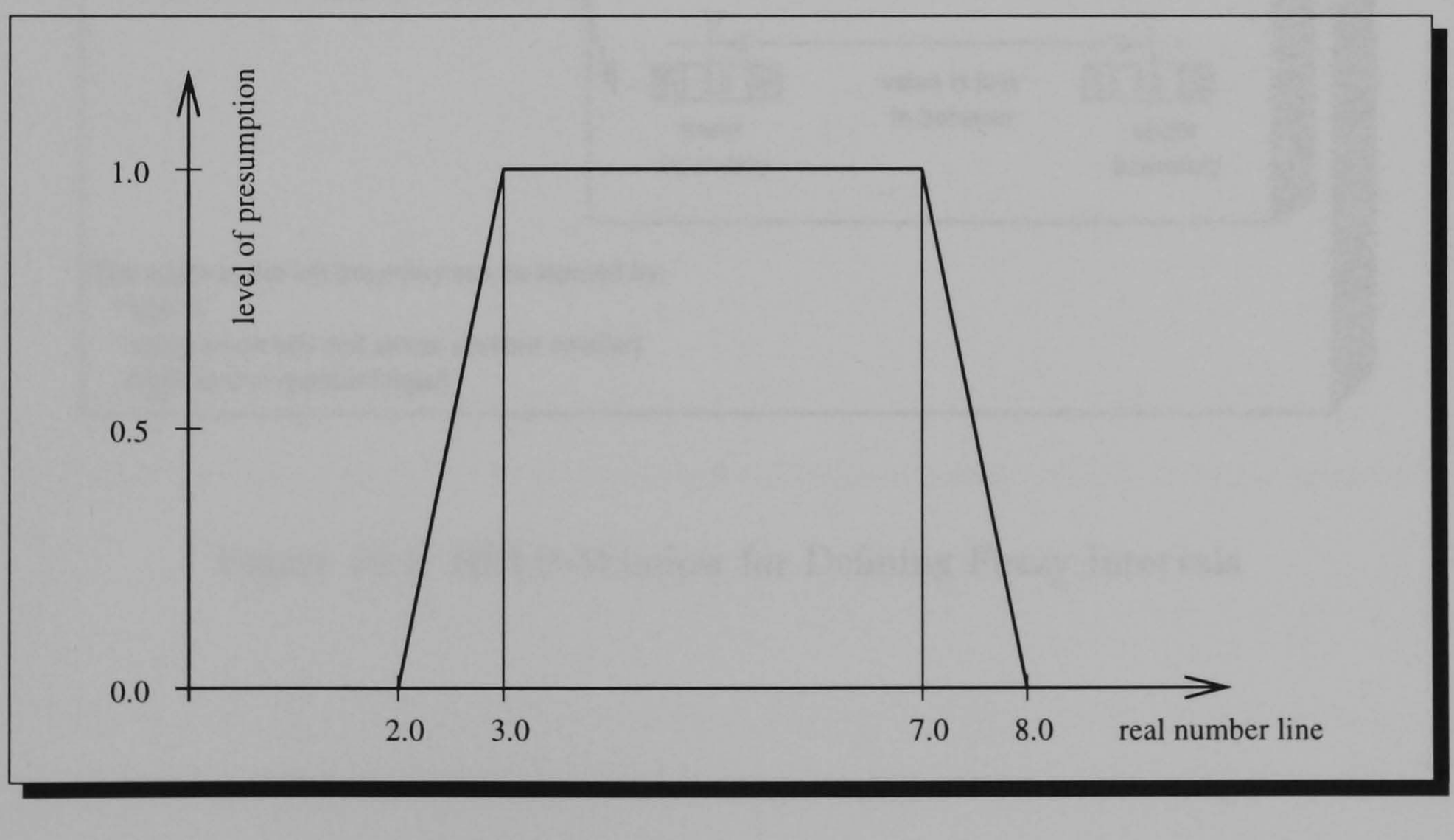


Figure 10.3: Membership Function of Vague Interval

The internal representation of the membership function Fig. 10.3 for the example

above (Fig.10.2) has linear left and right boundaries. The linear boundaries are used because of the simplicity of further computation. Since the membership function is defined by pure intuition of the designer, it makes no sense at this point to use membership functions which might be more suitable for the domain of electronic circuit design but need more computational number crunching.

A HELP-Window for the user interface (Fig.10.2) defining imprecise intervals is represented in Fig.10.4.

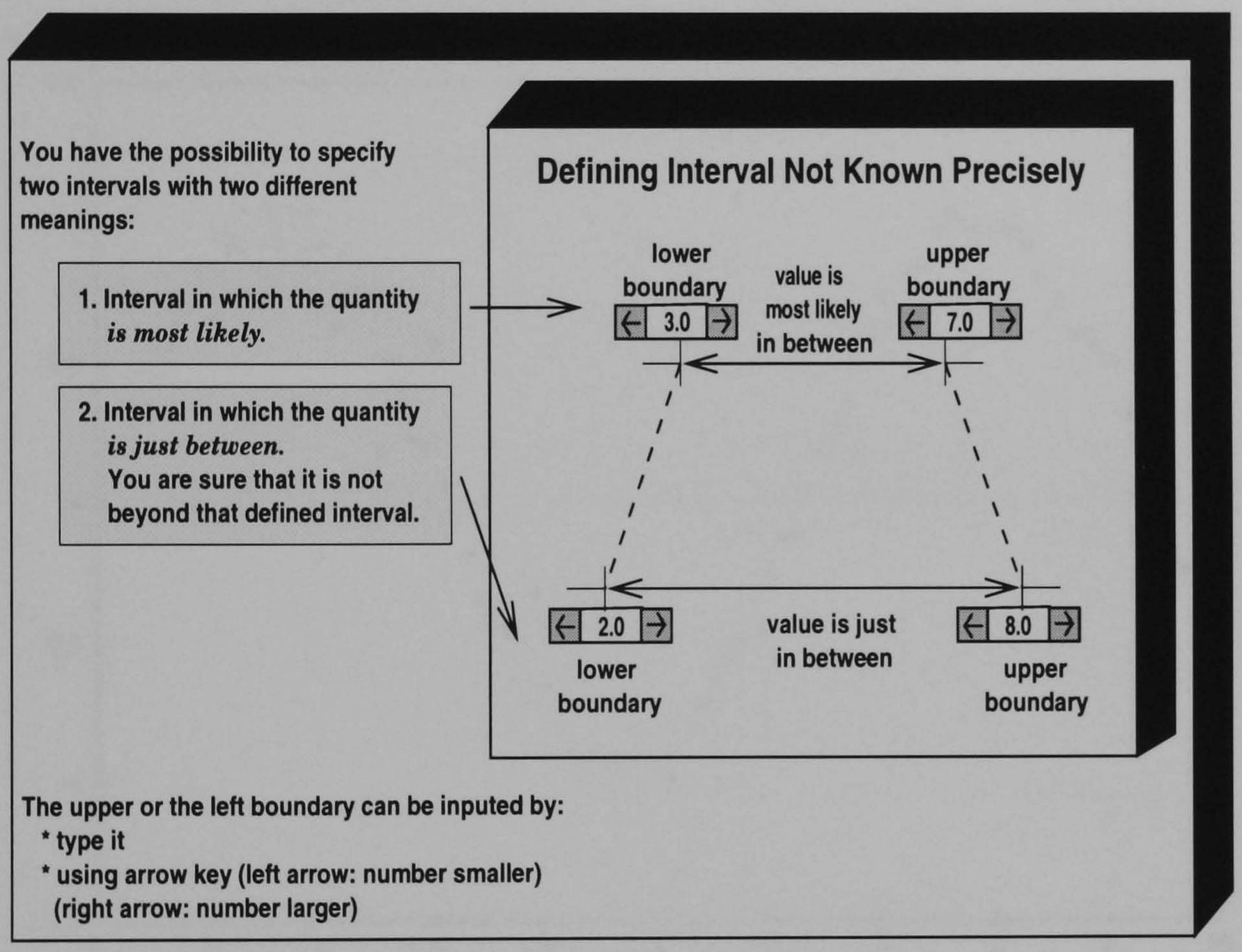


Figure 10.4: HELP-Window for Defining Fuzzy Intervals

10.3.3 Fuzzy Functions Model Vague Dynamic Input Data

As well as the time-dependent functional data (time series) as shown in Fig.10.5, there are temperature-dependent, frequency-dependent, etc., data. Generally data-sheets, electronic circuit design text books, and electronic designers visualize n-dimensional ($n = 2, 3, ...$) data using 2-dimensional graphs. Humans grasp more information looking at data visualized by graphs then by tabulated data. Suppose the system to be designed is driven by the input data in (Fig. 10.5):

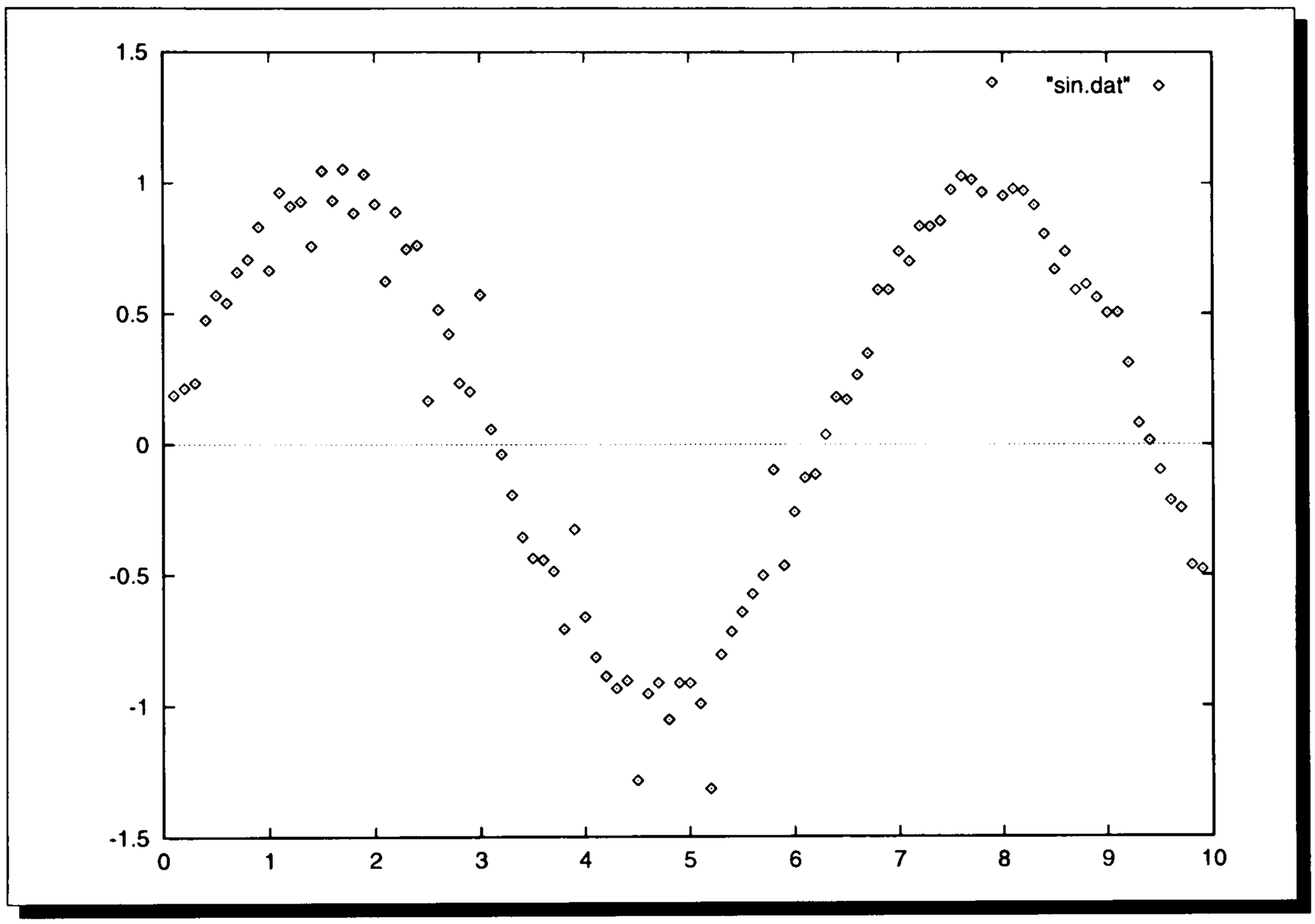


Figure 10.5: Measured Sinusoidal Signal

This signal (Fig.10.5) is a measured function which is easily approximated by a set of sine functions. This set of sine functions are used to represent this signal using the fuzzy approach by fuzzifying functions. The ill-defined time series can be specified by two pair of functions. A designer has to define an upper and a lower boundary function which has to be considered as most likely that the measurement

samples are in between. Further an upper and a lower boundary function could be defined which expresses the border at which the measurement samples are not beyond.

Similar to the fuzzy interval definition the following user interface is defined (Fig.10.6) for the definition of functions not known precisely.

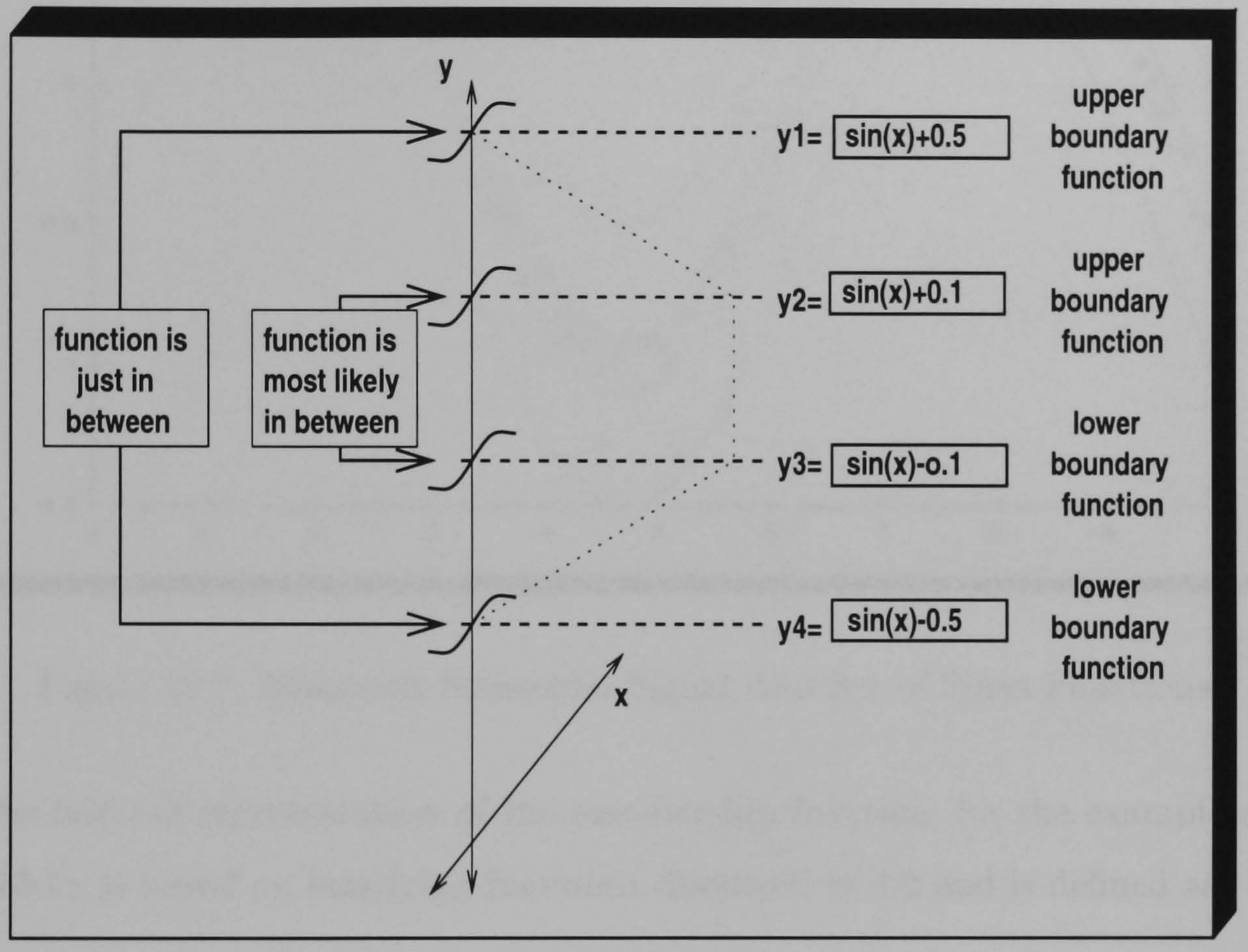


Figure 10.6: User Interface for Defining Vague Functions

After defining the set of functions (Fig.10.6), Fig.10.7 shows the measurement samples and their boundary functions. The inner pair of function has a membership value of 1.0 and represents the tube in which most of the measurement samples are. The outer pair of function has the membership value of > 0.0 and represents the tube in which no measurement sample lies outside.

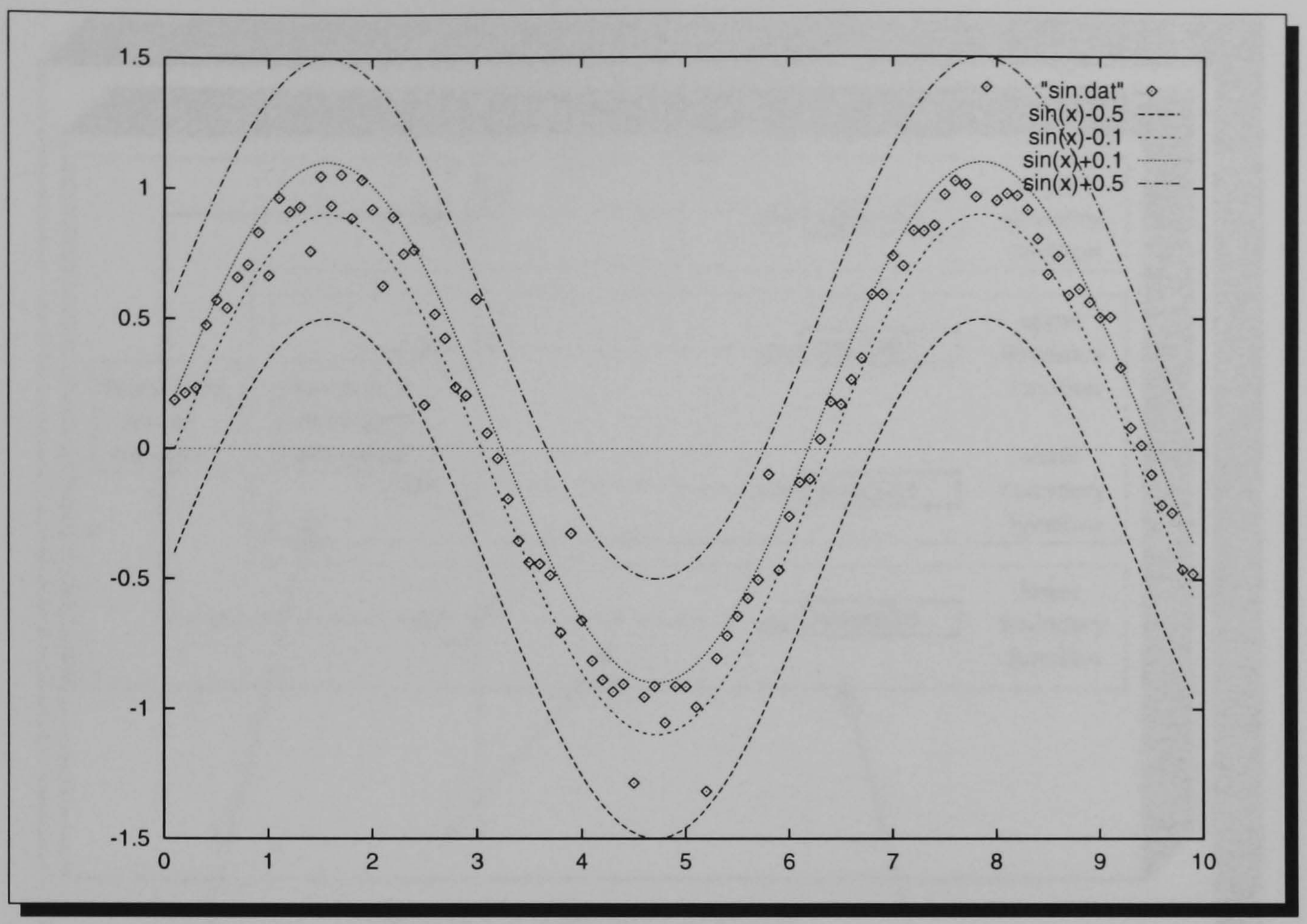


Figure 10.7: Measured Sinusoidal Signal And Set of Sinus Functions

The internal representation of the membership function, for the example above (Fig.10.7), is based on fuzzifying functions discussed in 4.2 and is defined as:

$$\mu_{\tilde{example}}(x,y) = \begin{cases} 0 & if \quad y \leq sin(x) - 0.5 \\ \frac{y-sin(x)-0.5}{0.4} & if \quad sin(x) - 0.5 < y \leq sin(x) - 0.1 \\ 1 & if \quad sin(x) - 0.1 < y < sin(x) + 0.1 \\ \frac{sin(x)+0.1-y}{0.4} & if \quad sin(x) + 0.1 \leq y < sin(x) + 0.5 \\ 0 & if \quad sin(x) + 0.5 \leq y \end{cases} \tag{10.1}$$

The contents of the HELP-Window (Fig.10.8) the designer gets from the system are for the user interface to define a fuzzy function.

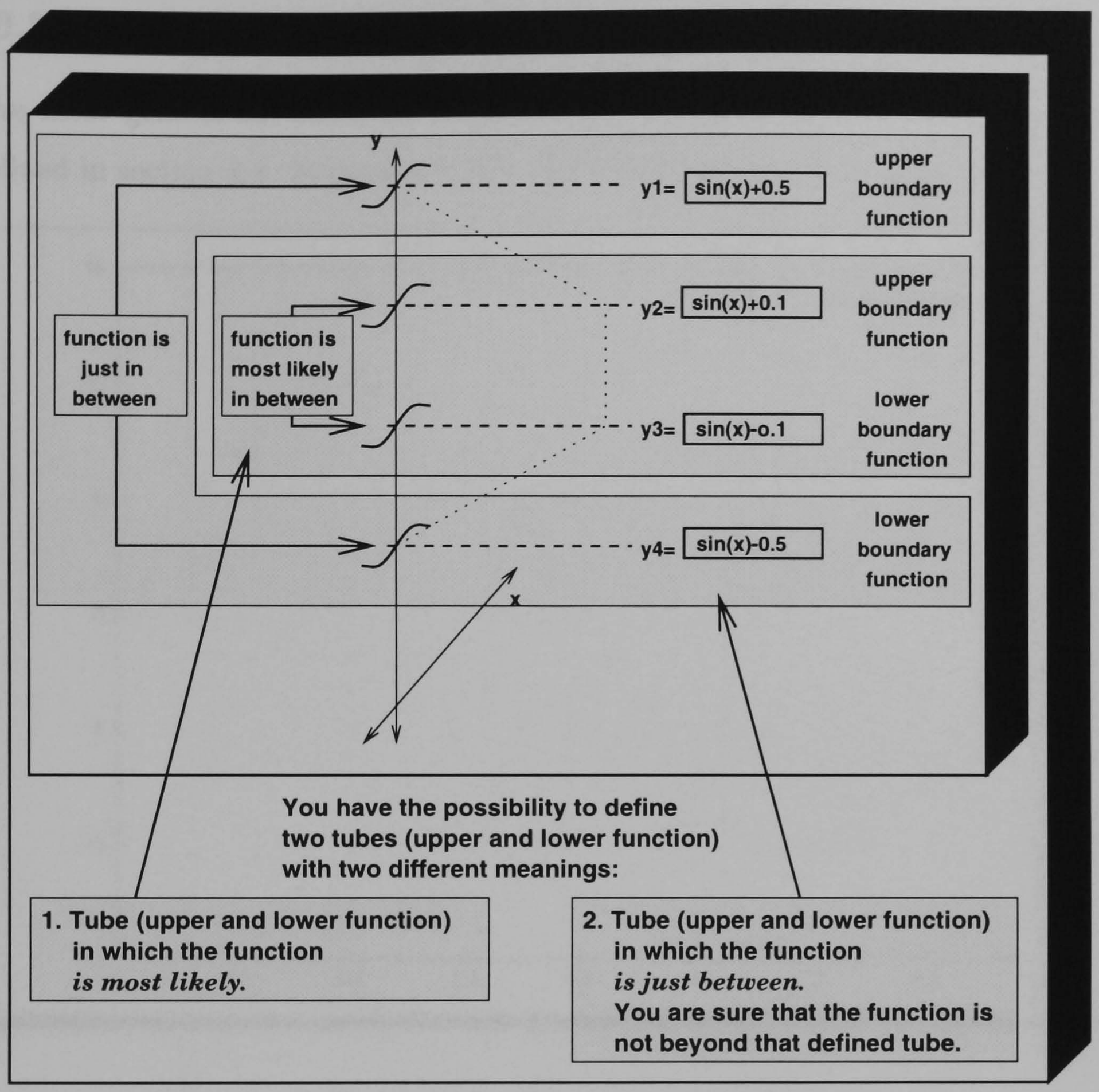


Figure 10.8: HELP-Window for Defining Vague Functions

10.3.4 Fuzzy Curves Model Vague Dynamic Input Data

The most general definition is, when an input is approximated by fuzzy curves defined in section 4.4. Suppose we have the following input data (Fig. 10.9):

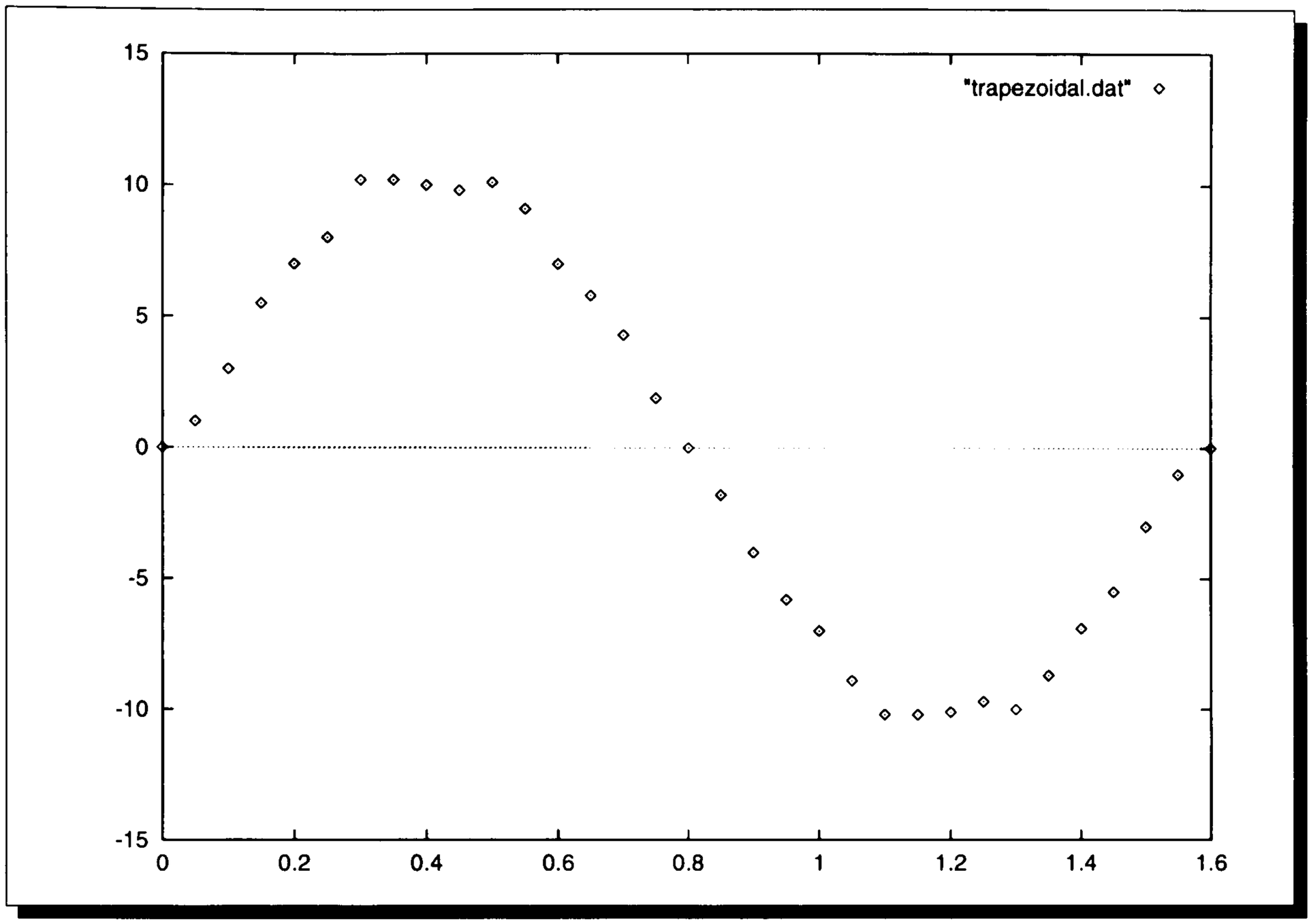


Figure 10.9: Trapezoidal Signal

This input signal is defined by the designer using a graph drawing tool. This tool allows to define the most likely signal section by linear approximated functions and a section in which the signal is just in between. Fig. 10.10 shows the user interface for the example above (Fig. 10.9) for defining such signals. To keep it simple we approximate the given input data by five sets of linear functions as show in Fig. 10.10.

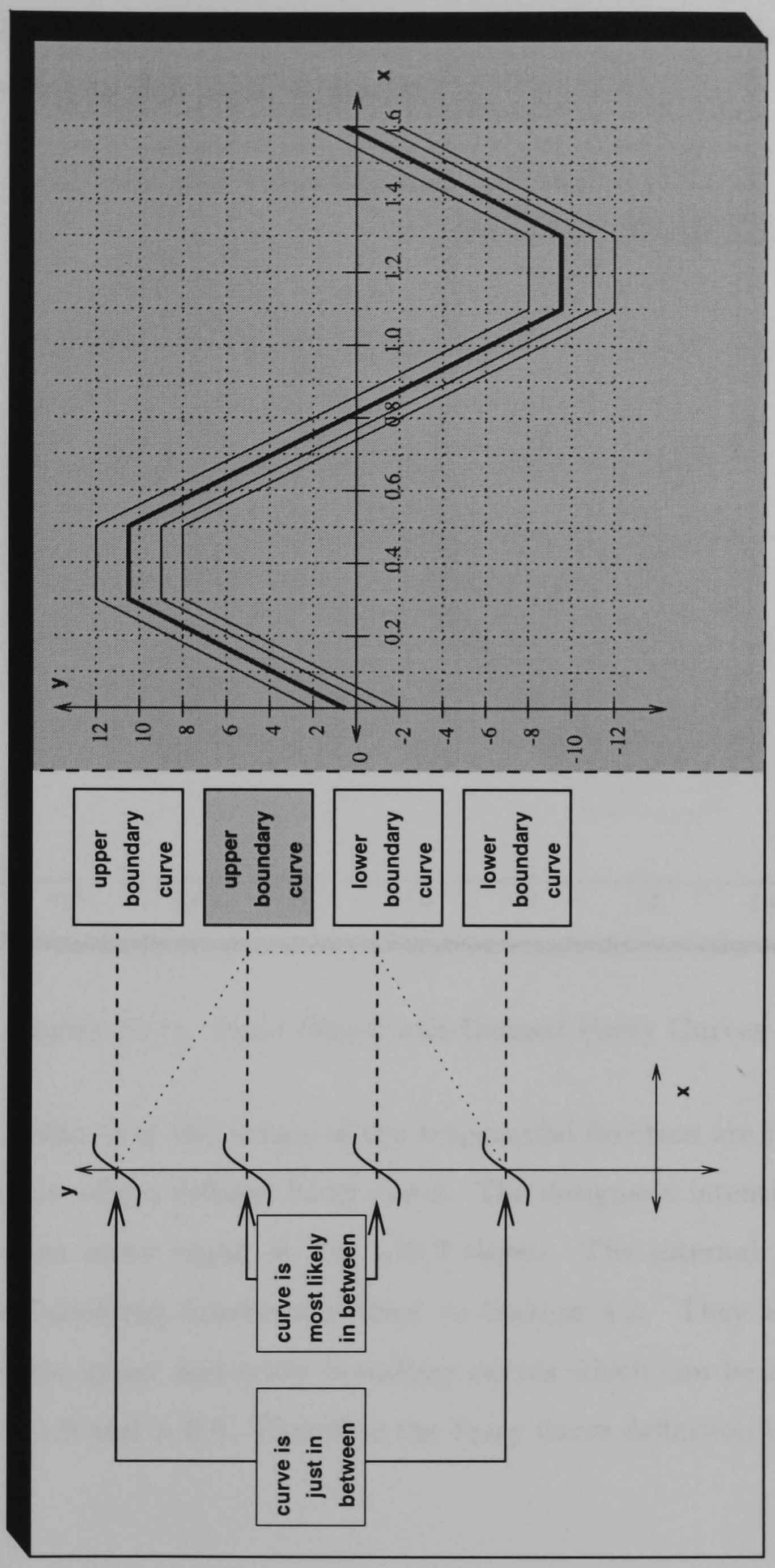


Figure 10.10: User Interface for Defining Fuzzy Curves

In Fig. 10.10 the *Upper Boundary Curve* is selected which indicates that the designer is working on that curve at present.

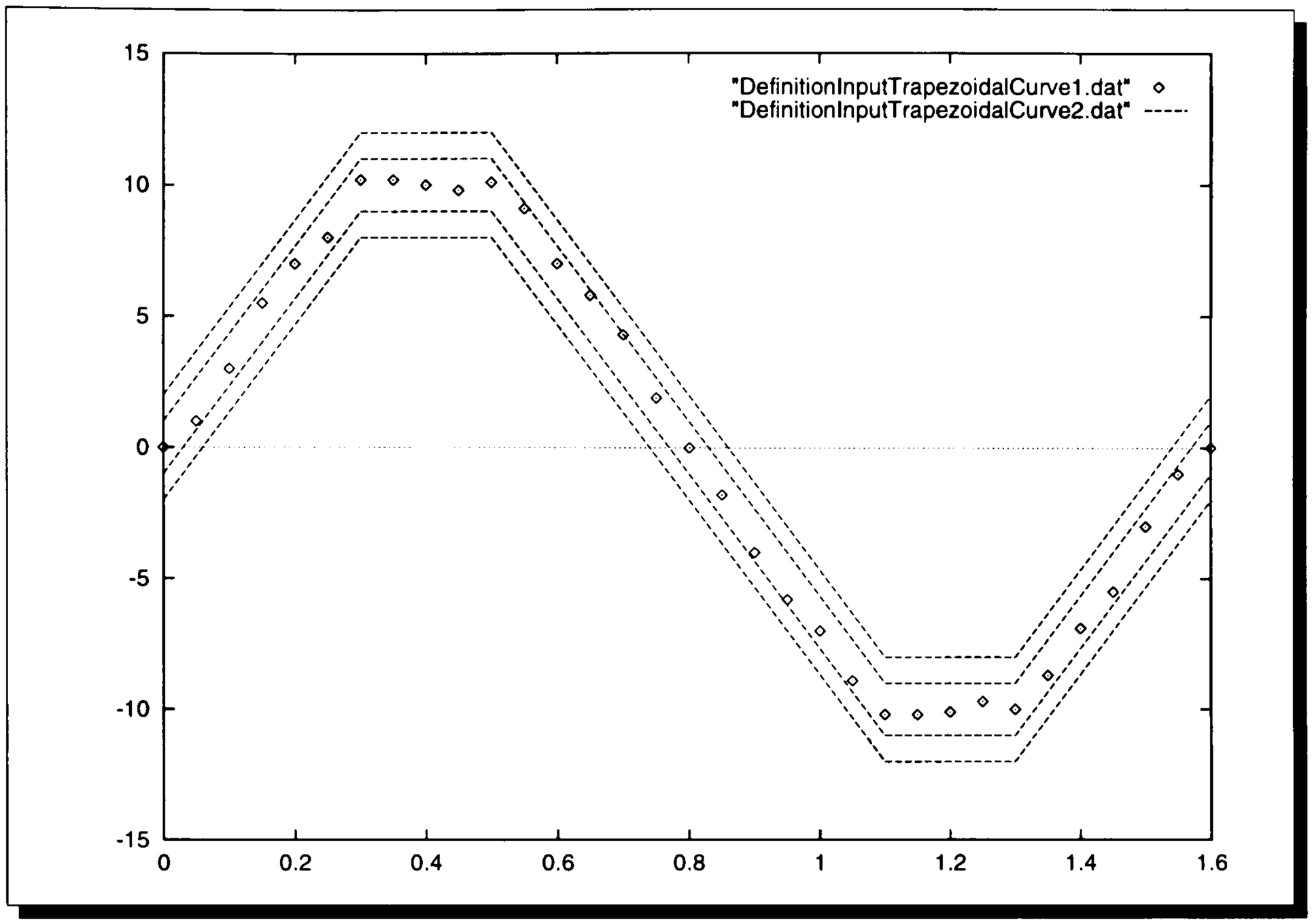


Figure 10.11: Input Signal and Defined Fuzzy Curves

Fig. 10.11 shows that the values of the trapezoidal function are in between the most likely section of the defined fuzzy curve. The designer's intention is that the input data is even more vague as Fig. 10.9 shows. The internal representation consists of five fuzzifying functions defined in Section 4.2. They are completely defined by the two upper and lower boundary curves which can be represented by the two α-levels 1.0 and > 0.0. Therefore the fuzzy curve definition is defined as:

$$\tilde{R}^1\text{:} \quad IF\ x\ is\ 0 \geq x \leq 0.3,\ THEN\ \tilde{y}\ is\ \tilde{f}_1(x)$$
$$\tilde{R}^2\text{:} \quad IF\ x\ is\ 0.3 > x \leq 0.5,\ THEN\ \tilde{y}\ is\ \tilde{f}_2(x)$$
$$\tilde{R}^3\text{:} \quad IF\ x\ is\ 0.5 > x \leq 1.1,\ THEN\ \tilde{y}\ is\ \tilde{f}_3(x)$$
$$\tilde{R}^4\text{:} \quad IF\ x\ is\ 1.1 > x \leq 1.3,\ THEN\ \tilde{y}\ is\ \tilde{f}_4(x)$$
$$\tilde{R}^5\text{:} \quad IF\ x\ is\ 1.3 > x \leq 1.6,\ THEN\ \tilde{y}\ is\ \tilde{f}_5(x)$$

Restricting it to fuzzy intervals and linear boundaries we get for the following membership function

for $\tilde{y} = \tilde{f}_1(x)$:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq 33.3 * x - 2 \\ y - 33.3 * x + 2 & if \quad 33.3 * x - 2 < y \leq 33.3 * x - 1 \\ 1 & if \quad 33.3 * x - 1 < y < 33.3 * x + 1 \\ 33.3 * x + 2 - y & if \quad 33.3 * x + 1 \leq y < 33.3 * x + 2 \\ 0 & if \quad 33.3 * x + 2 \leq y \end{cases}$$

for $\tilde{y} = \tilde{f}_2(x)$:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq 8 \\ y - 8 & if \quad 8 < y \leq 9 \\ 1 & if \quad 9 < y < 11 \\ 12 - y & if \quad 11 \leq y < 12 \\ 0 & if \quad 12 \leq y \end{cases}$$

for $\tilde{y} = \tilde{f}_3(x)$:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq -33.3 * x + 25.7 \\ y + 33.3 * x - 25.7 & if \quad -33.3 * x + 25.7 < y \leq -33.3 * x + 24.7 \\ 1 & if \quad -33.3 * x + 24.7 < y < -33.3 * x + 23.7 \\ 21.7 - 33.3 * x - y & if \quad -33.3 * x + 22.7 \leq y < -33.3 * x + 21.7 \\ 0 & if \quad -33.3 * x + 21.7 \leq y \end{cases}$$

for $\tilde{y} = \tilde{f}_4(x)$:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq -12 \\ y + 12 & if \quad -12 < y \leq -11 \\ 1 & if \quad -11 < y < -9 \\ -9 - y & if \quad -9 \leq y < -8 \\ 0 & if \quad -8 \leq y \end{cases}$$

for $\tilde{y} = \tilde{f}_5(x)$:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad y \leq 33.3 * x - 55.3 \\ y - 33.3 * x + 55.3 & if \quad 33.3 * x - 55.3 < y \leq 33.3 * x - 54.3 \\ 1 & if \quad 33.3 * x - 54.3 < y < 33.3 * x - 53.3 \\ 33.3 * x - 53.3 - y & if \quad 33.3 * x - 53.3 \leq y < 33.3 * x - 52.3 \\ 0 & if \quad 33.3 * x - 52.3 \leq y \end{cases}$$

The contents of the HELP-Window for the user interface to define a fuzzy curve is shown in Fig.10.12.

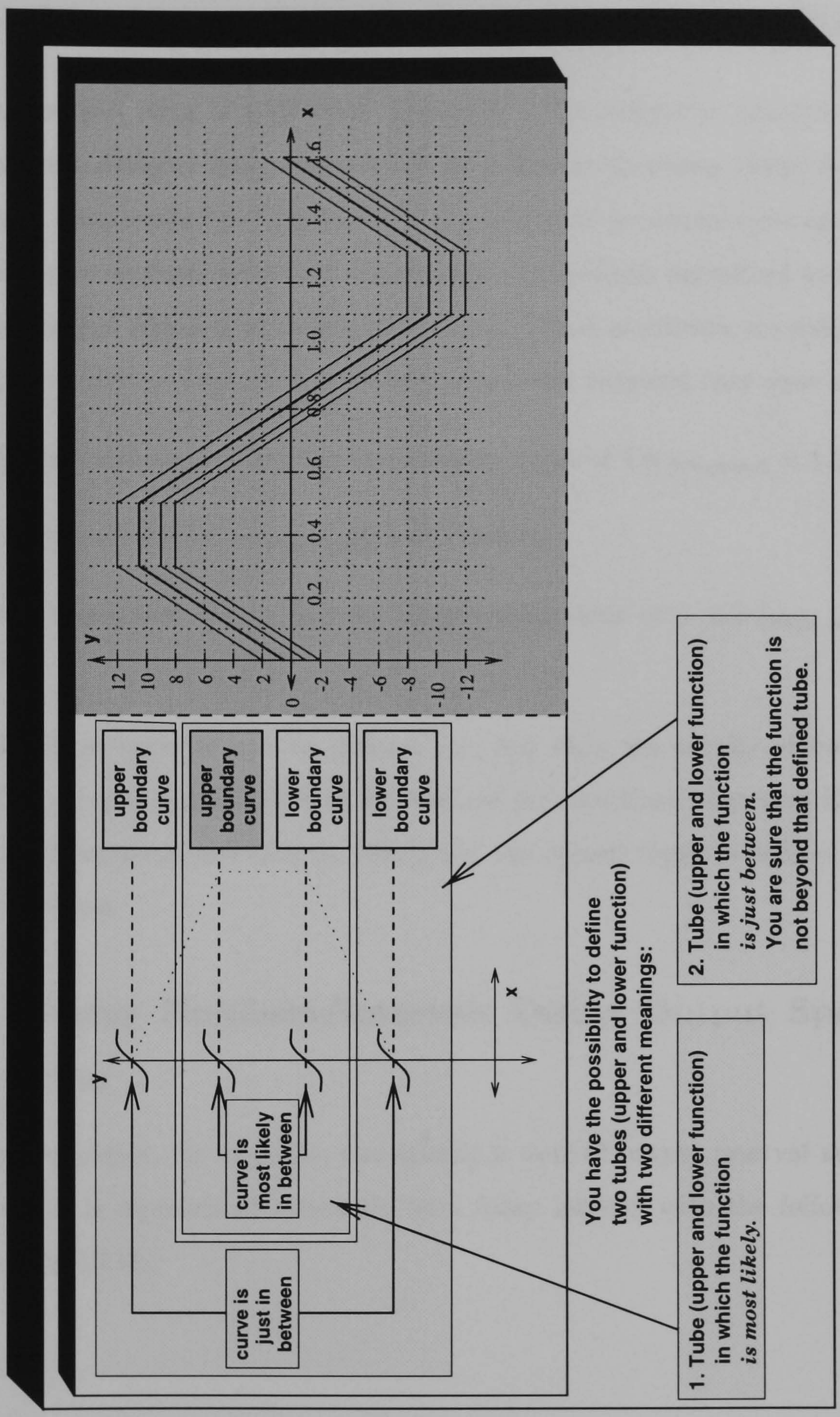


Figure 10.12: HELP-Window for Defining Vague Curves

10.4 Specifying Output Data Not Known Exactly

Specifying output data is a different approach. The output is generated by the system and the designer has to compare it with the specifications (what should be). The output data could be categorized into sections of permissiveness used in this work. There are sections where the output signal is optimal, permitted and sections where the output signal is not permitted at all. These attributes are mapped to a level of presumption of the membership function and adjusted only once a session.

- The **optimal** section has the membership value of 1.0 ($\mu_{optimal} = 1.0$),
- the **allowed** section is $\mu_{permitted} = 0.5$ and the
- **not permitted** section has the membership value of > 0.0 ($\mu_{not\ permitted} > 0.0$).

All the levels in between can be defined too, but they are usually of no interest. Generally the input signal is defined by levels of presumptions (stated at Kaufmann and Gupta [Kaufmann and Gupta, 1991]) and the output signal is defined by levels of permissiveness.

10.4.1 Fuzzy Numbers/Intervals Define Output Specification

The design engineer, for example, can specify a wanted output interval not known exactly which is represented internally by a fuzzy interval with the following user interface (Fig.10.13):

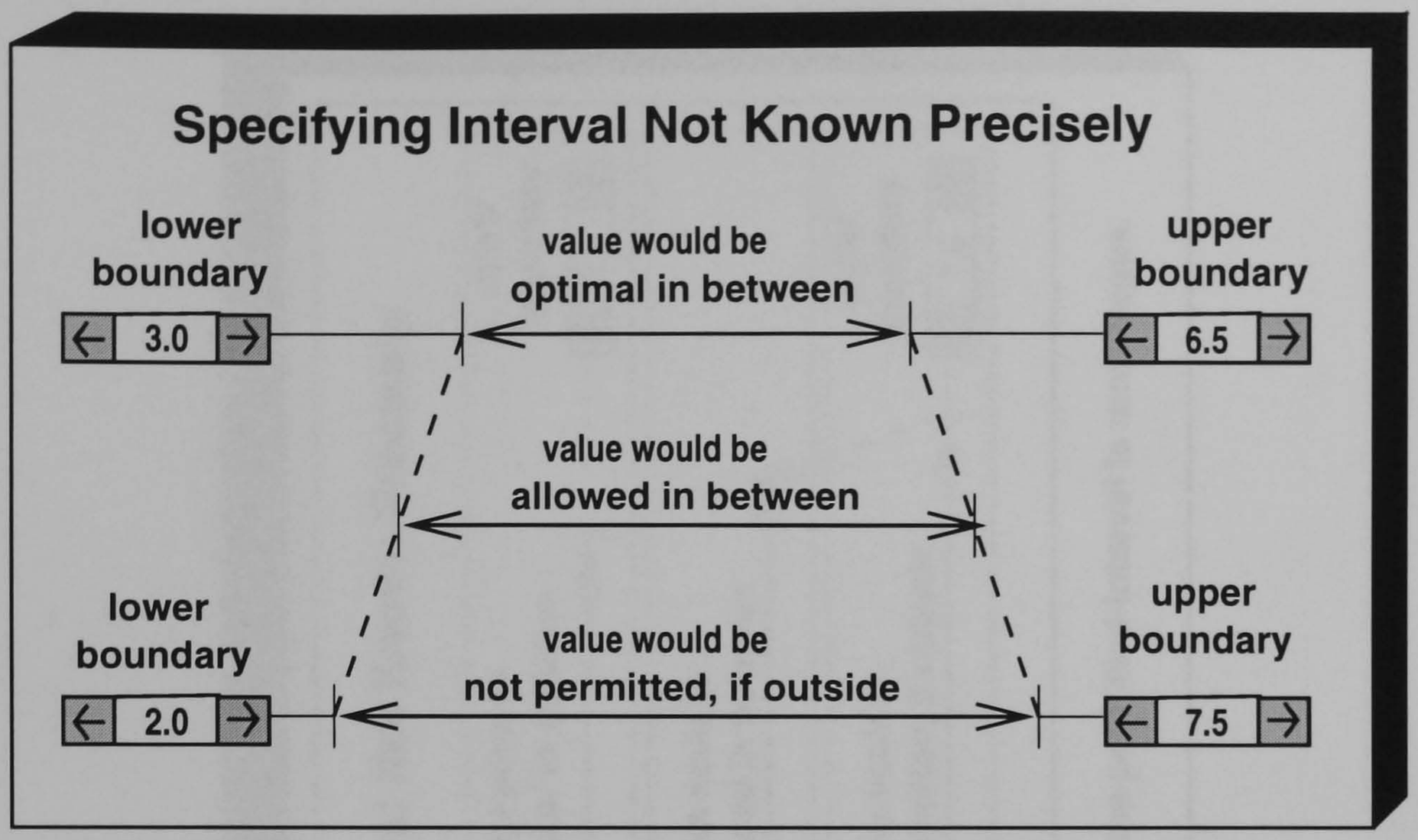


Figure 10.13: User Interface to Specify a Wanted Output Interval

The contents of the HELP-Window (Fig.10.14) the user gets from the system:

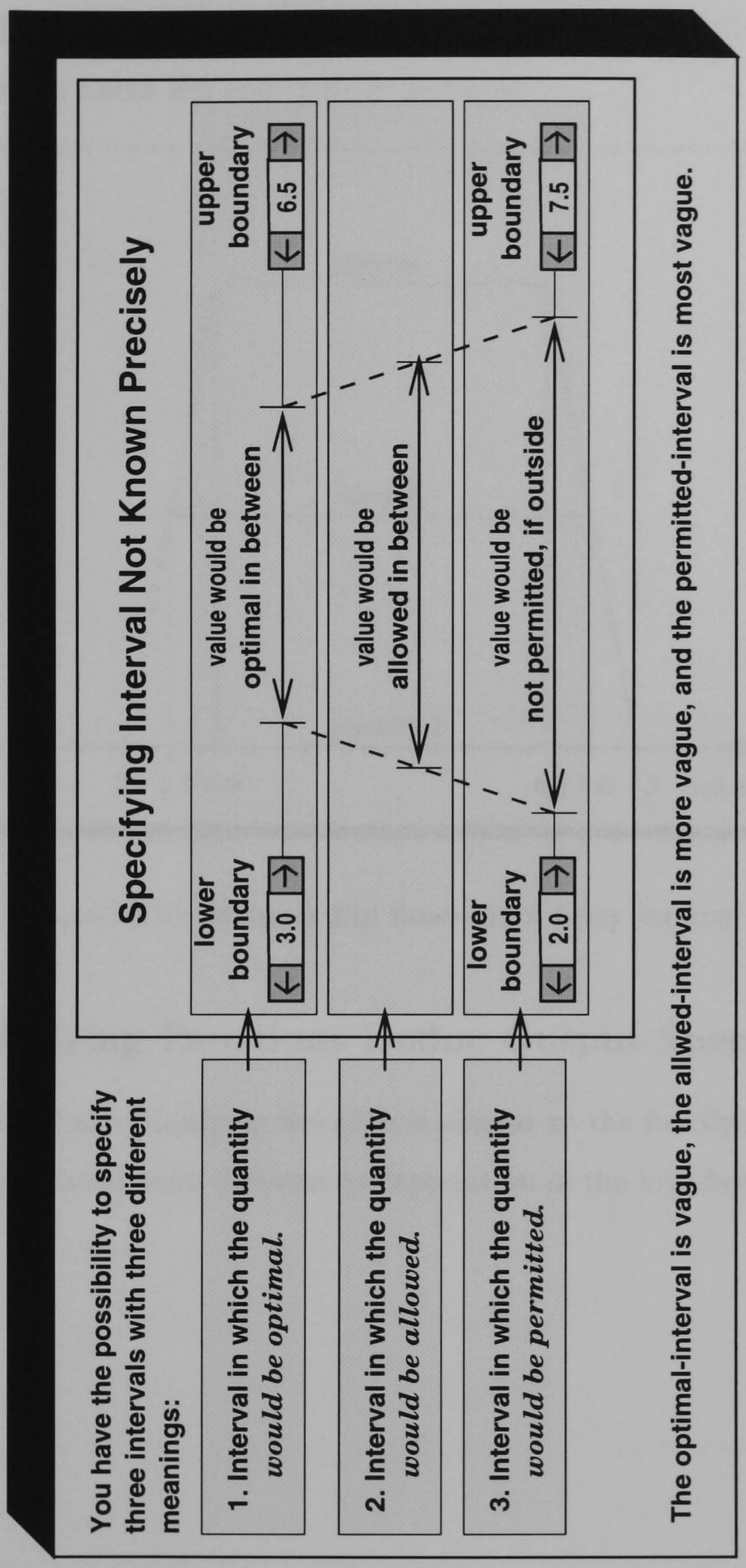


Figure 10.14: HELP-Window for specifying a wanted output interval

The internal representation of the membership function Fig.10.15 for the example Fig.10.13 above has linear left and right boundaries:

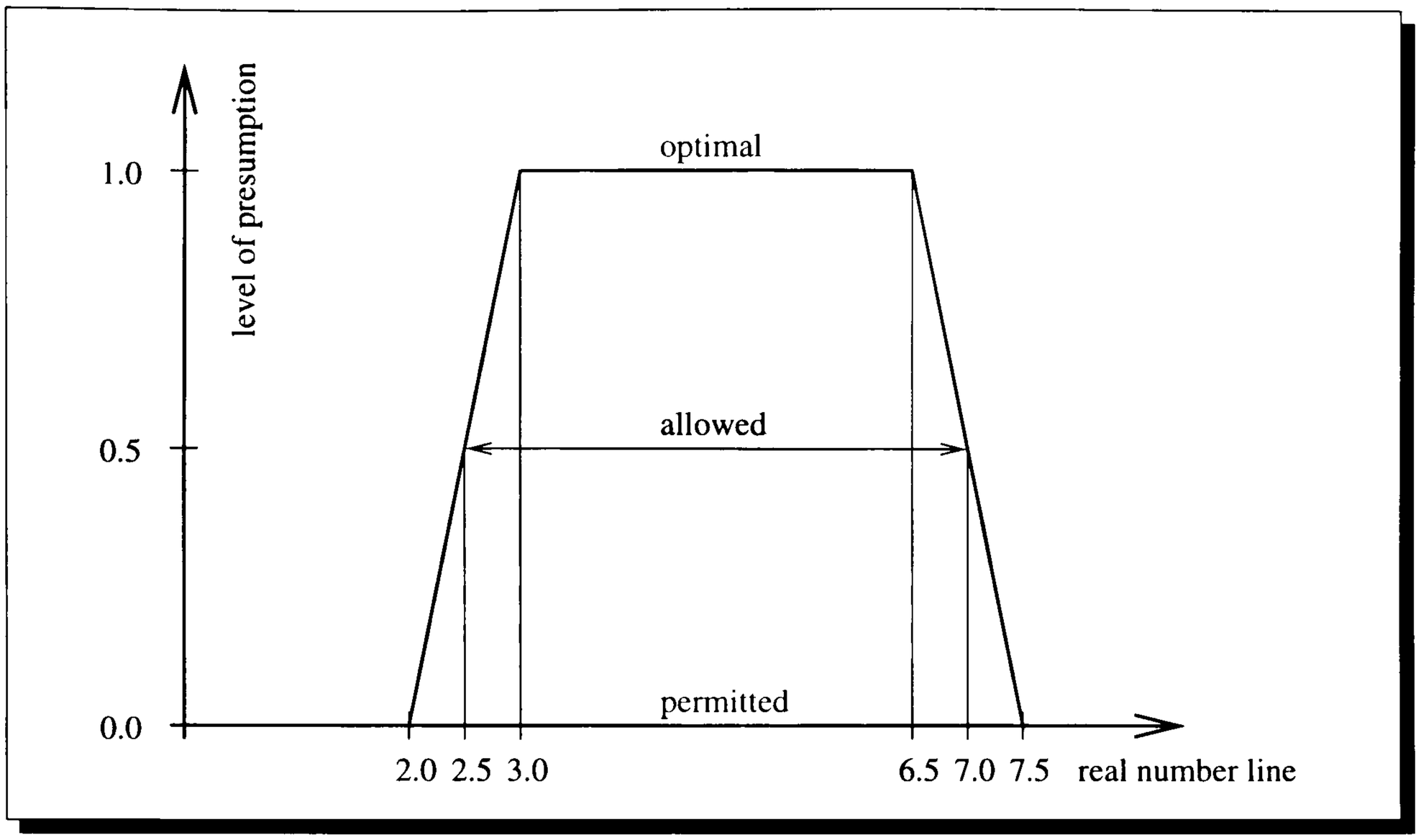


Figure 10.15: Membership function of fuzzy interval

10.4.2 Fuzzifying Functions Define Output Specification

The specification of the fuzzifying function is similar to the fuzzifying definition of vague input data except with different interpretation of the membership function.

10.4.3 Fuzzy Curves Define Output Specification

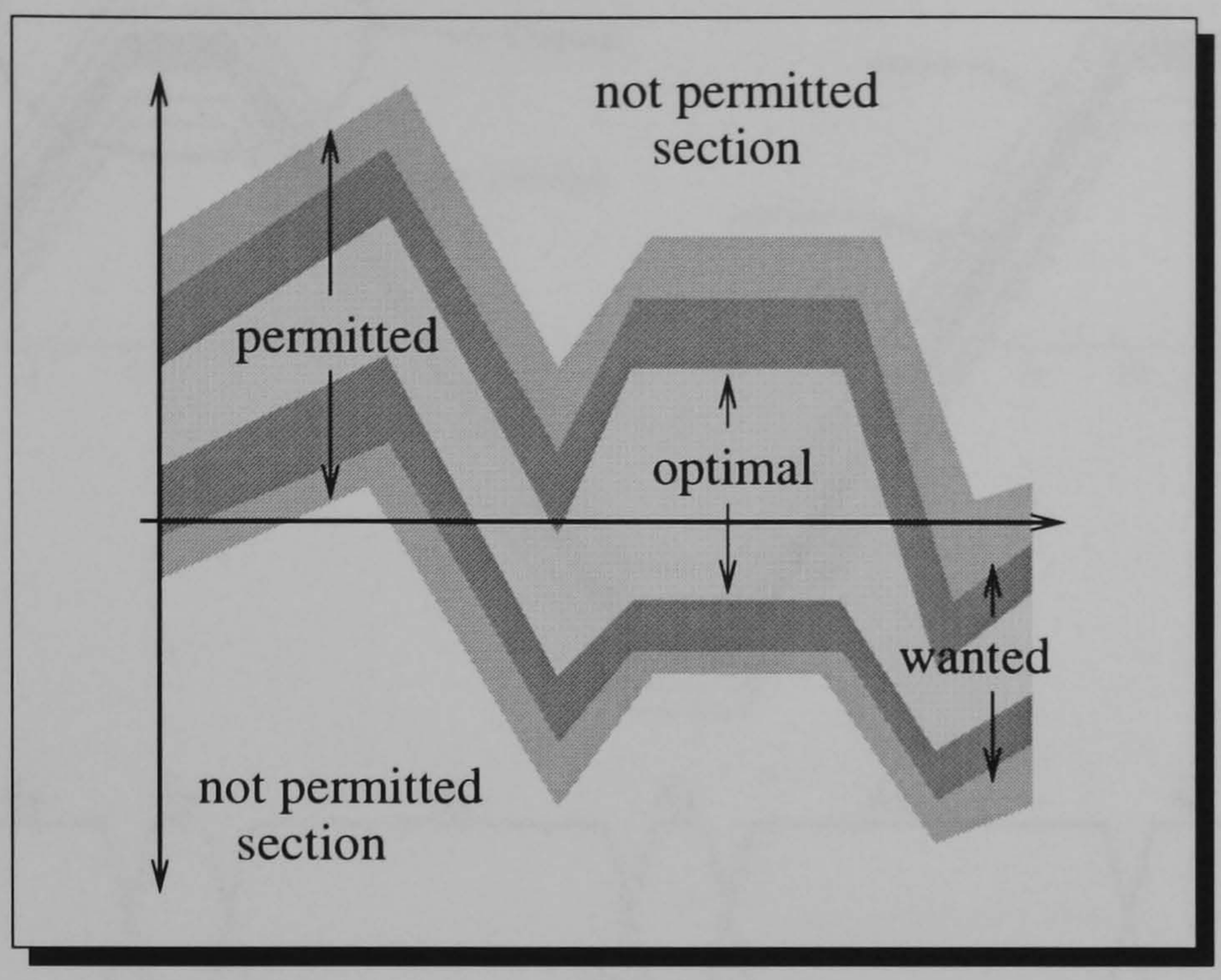


Figure 10.16: Fuzzy Output Signal Specification

A more general example illustrates Fig. 10.16. It shows the regions where an output signal is optimal, wanted, and permitted. Fig. 10.16 could be represented by fuzzy curves as introduced in section 4.4.

Example: Output Specification by Fuzzy Curve

Suppose we want to specify a trapezoidal output signal (Fig.10.17) in sections of permissions optimal, wanted and permitted.

The grey section of Fig.10.17 specifies where the signal would be optimal. The sections beyond the upper and lower line specifies where the signal is not permitted. The fuzzy intervals for the x-axes define how precisely we know the time intervals.

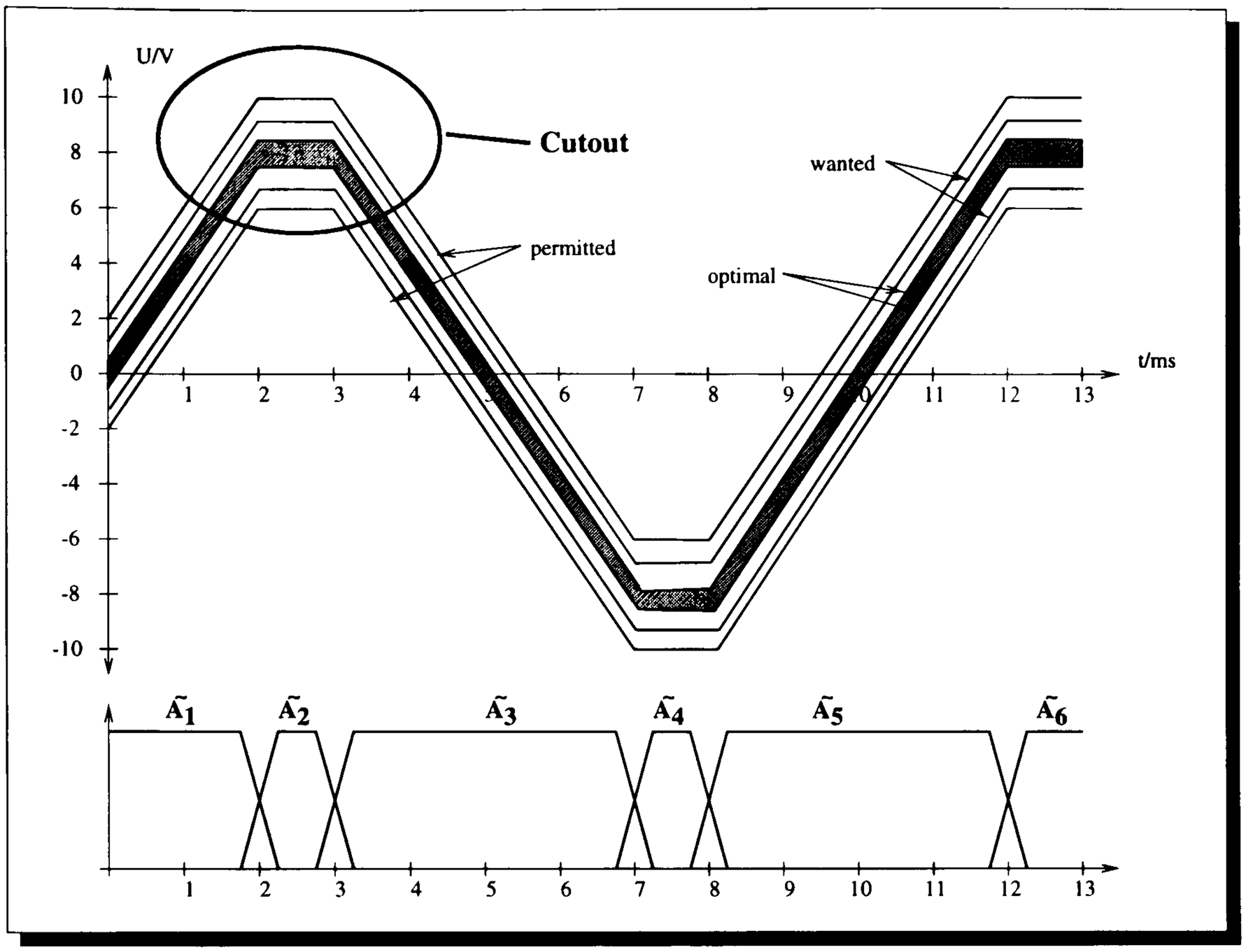


Figure 10.17: Fuzzy Trapezoidal Output Signal

The piecewise represented signal specification is interrupted. At the transitions the membership value is determined as shown in the cutout (Fig.10.18) of the Fuzzy Relation Memory.

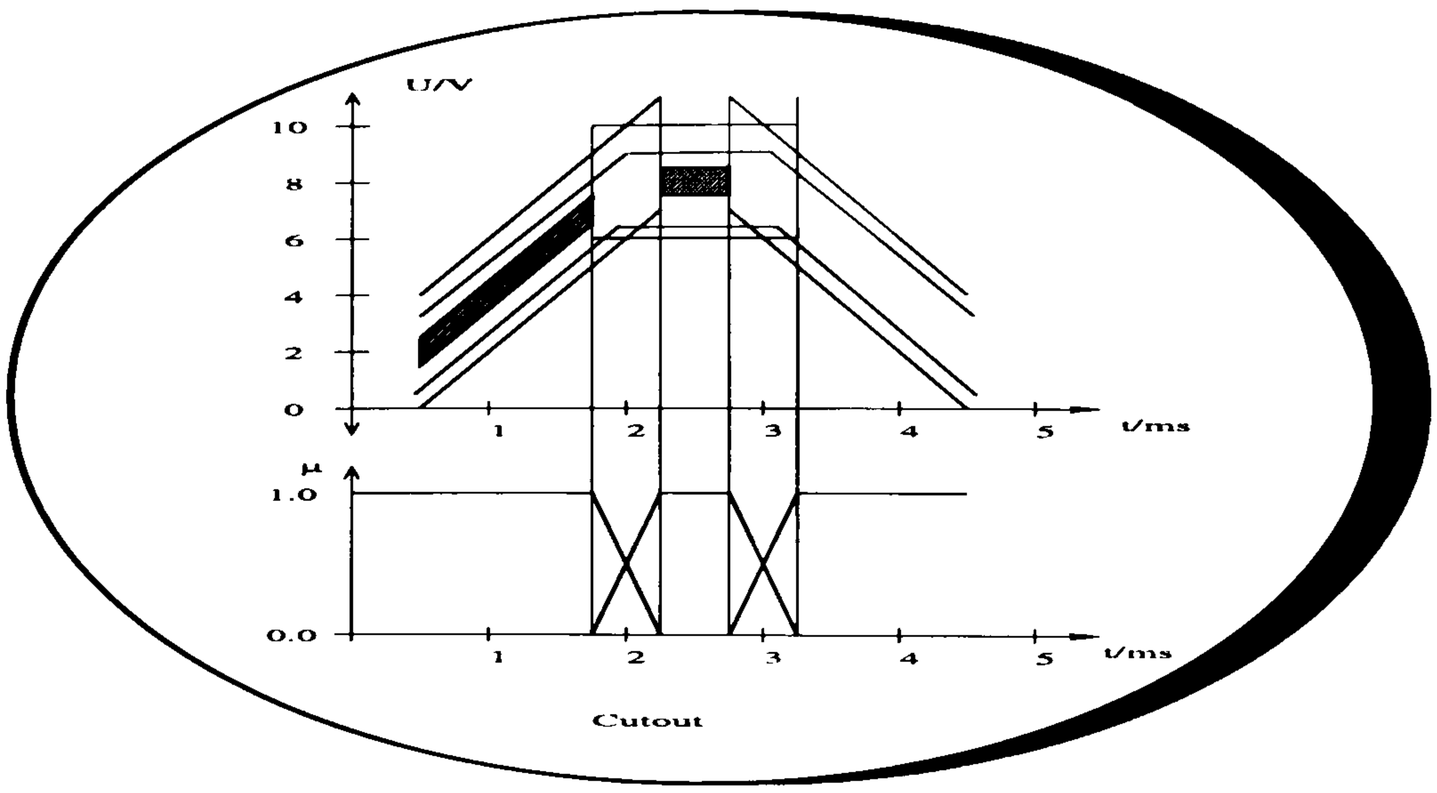


Figure 10.18: Cutout of Fuzzy Trapezoidal Output Signal

The internal representation of FRM of this fuzzy defined signal is the canonical rule-based form with 6 relations. Relation 2 with the fuzzy interval $\tilde{A}_2$ is defined as follows:

$$\tilde{R}^2\text{:} \quad IF\ x\ is\ \tilde{A}_2,\ THEN\ \tilde{y}\ is\ \tilde{f}_2(x)$$

The membership function of $\tilde{A}_2$ is defined, for example:

$$\mu_{\tilde{A}_2}(x) = \begin{cases} 0 & if \quad x \leq 1.8 \\ \frac{x-1.8}{0.4} & if \quad 1.8 < x \leq 2.2 \\ 1 & if \quad 2.2 < x < 2.8 \\ \frac{3.2-x}{0.4} & if \quad 2.8 \leq x < 3.2 \\ 0 & if \quad 3.2 \leq x \end{cases} \tag{10.2}$$

The membership function of $\tilde{y}$ is defined, for example:

$$\mu_{\tilde{y}}(x,y) = \begin{cases} 0 & if \quad x \leq 6.0 \\ \frac{y-6.0}{1.0} & if \quad 6.0 < y \leq 6.5 \\ \frac{y-5.5}{2.0} & if \quad 6.5 < y \leq 7.5 \\ 1 & if \quad 7.5 < y < 8.5 \\ \frac{10.5-y}{2.0} & if \quad 8.5 \leq y < 9.5 \\ \frac{10-y}{1.0} & if \quad 9.5 \leq y < 10 \\ 0 & if \quad 10 \leq y \end{cases} \tag{10.3}$$

and independent of x.

10.5 Modelling

As stated in Chapter 7 there are three kinds of modelling possible:

1. imprecise functional-based modelling
2. imprecise rule-based modelling
3. imprecise constraint-based

Rule-based and constraint-based modelling are defined by IF-THEN rules and differential equations respectively. For functional-based modelling the model is created by connecting basic functional blocks visualized by symbols (see Section 10.5.1).

The modelling approach allows the description of dynamic physical systems by different kind of behavioural models. Features of behavioural modelling are:

- modelling at different levels of abstraction is allowed, e.g. device (circuit) level, functional level, signal flow level, etc.
- flow control statements (if-else, loops)
- mathematical functions

- flexible specifications of voltage/current relationships
- model implementation does not have to reflect hardware implementation. The designer does not need to know the physical implementation of a device to write a behavioural model.
- A behavioural model is as accurate as the equations used, i.e. it does not have to be less accurate than a device level model.

10.5.1 Symbols for Modelling

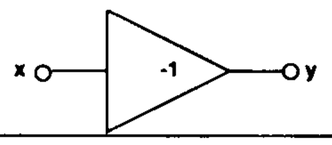
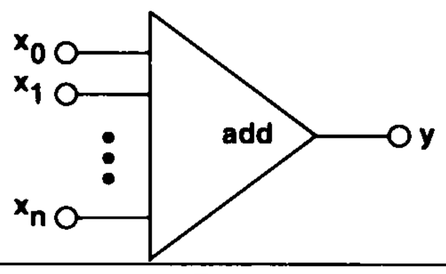
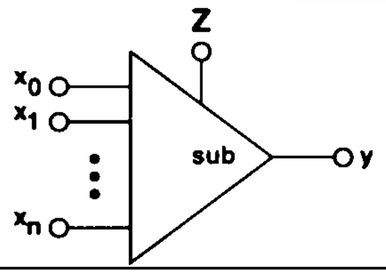
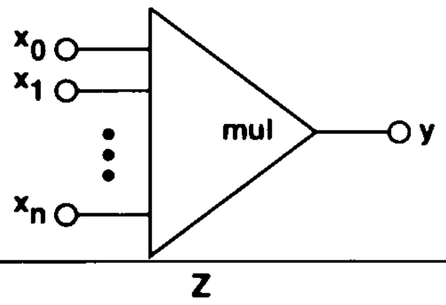
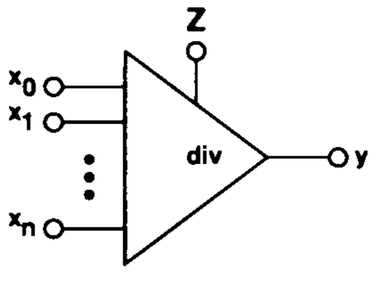
Name	Symbol	Function
inverter		$y = -x$
addition[1]		$y = \sum_{i=0}^{n} x_i$
subtraction[1]		$y = Z - \sum_{i=0}^{n} x_i$
multiplication[1]		$y = x_0 * x_1 * \ldots * x_n$
division[1]		$y = \frac{Z}{x_0 * x_1 * \ldots * x_n}$

Figure 10.19: Symbols of the Paper Prototype A

[1] non-interactive, strongly positive interactive, or strongly negative interactive definable

Name	Symbol	Function

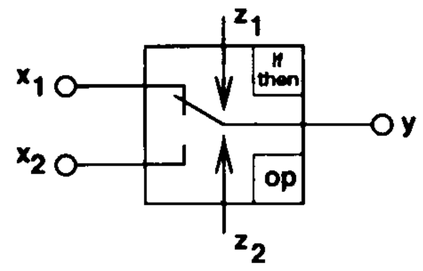
if-then		*if* z_1 **op** z_2 *then*
		$y = x_2$
		else
		$y = x_1$

Figure 10.20: Symbols of the Paper Prototype B

10.5.2 Hierarchy — Important Concept in Modelling

Hierarchical design is a very important concept. In order to avoid combinatorial explosions the most abstract description that will suffice to solve a problem is to be used. Starting with an abstract description and resorting to a detailed description only when necessary, irrelevant problem-solving work can be avoided and the complexity of analogue circuit design can be handled. Hierarchical design is the possibility to break the overall design task in more manageable subtasks, which are again broken up in smaller subtasks, etc. The division into more manageable abstract descriptions is arbitrary but should be descriptions which are easily represented by realizable blocks. Abstract descriptions are distinguished by the individual designer and mostly a small functional part of the overall function.

10.5.3 Example: Functional-Based Model of an Operational Amplifier

A designer wants to design a comparator circuit, that indicates when a signal is greater than a particular voltage. Fig. 10.21 shows the model of such a comparator using functional-based modelling.

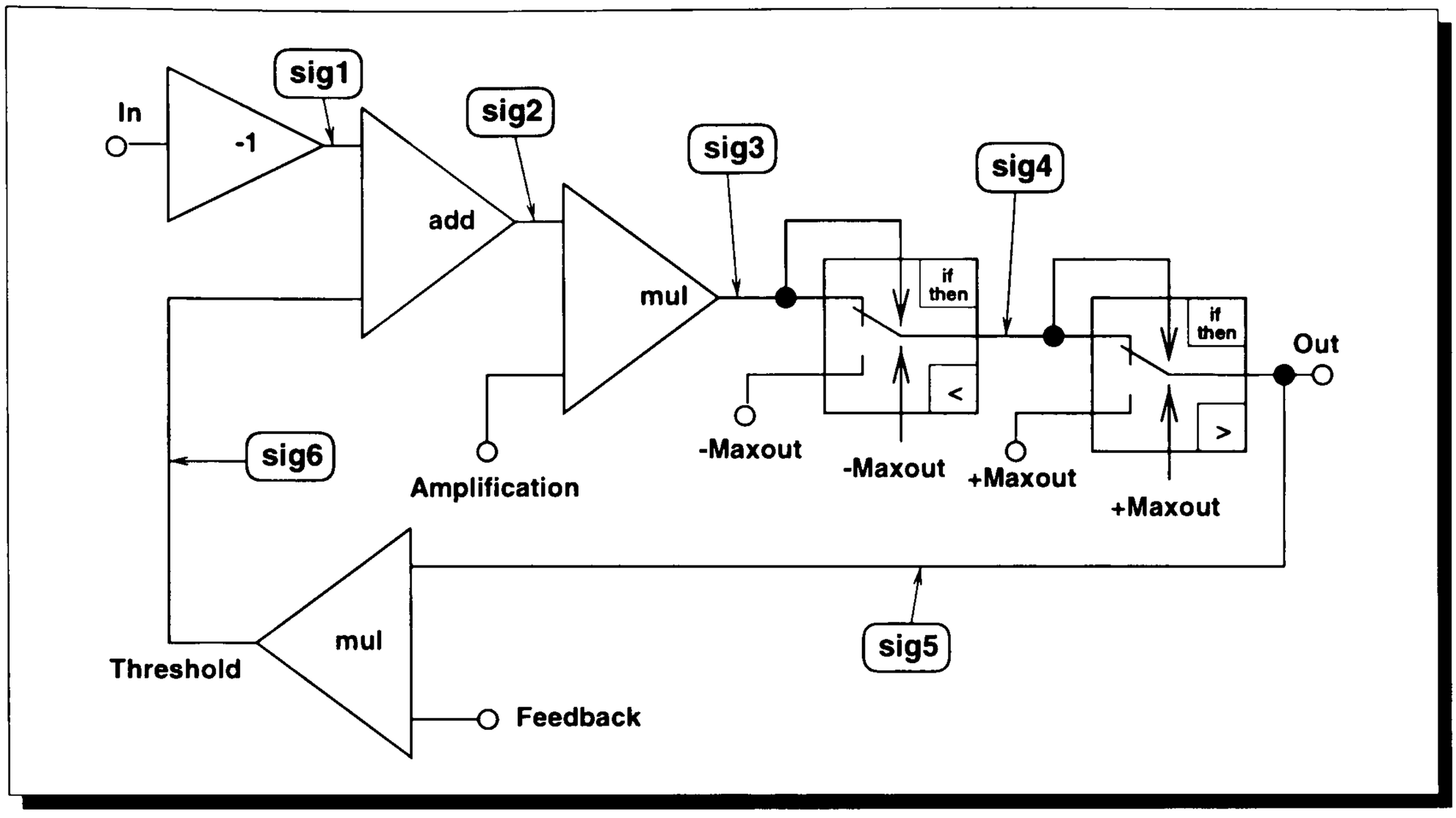


Figure 10.21: Model of a Comparator

Internally the model is represented by a set of rules and equations:

- $sig1 = -In$
- $sig5 = Out$
- $sig6 = sig5 * Feedback$
- $sig2 = sig1 + sig6$
- $sig3 = sig2 * Amplification$
- IF $sig3 < -Maxout$ THEN $sig4 = -Maxout$
- IF $sig3 \geq -Maxout$ THEN $sig4 = sig3$
- IF $sig4 > +Maxout$ THEN $sig5 = +Maxout$
- IF $sig4 \leq +Maxout$ THEN $sig5 = sig4$

10.6 Conclusion

Most important for an analogue circuit design tool used by design engineers is to follow their way of describing analogue circuits. The collection of user interfaces shown in this chapter support the designer by the definition of imprecise system specifications. Especially to overcome the difficulties in defining the membership function of fuzzy values implicitly done using the user interfaces. The intuition of the circuit designer defines the membership functions. Of advantage is that the person developing the membership functions can easily interpret the result of a simulation. Problematic is that other designers might have difficulties or interpret results wrongly because they have a different understanding of the membership functions.

Chapter 11

Circuit Paper Prototype Design

This chapter discusses the high-level design of a **Circuit Paper Prototype Design Phase**. It explains what high-level design means and argues that Fuzzy Relation Memory (Fuzzy Curves) described in Section 4.4 are a step towards graph understanding. For the Circuit Paper Prototype Design Phase it is explained how models are tested. Summarized is this chapter giving a detailed example showing qualitative and fuzzy simulation at the circuit paper prototype design level.

11.1 High-level Design

The circuit paper prototype is designing at a high-level, where ideas and various experimental prototypes can be investigated. Important features at this level of design are:

- Allow designers to draw a model of an analogue circuit using a special-purpose graphical editor that provides domain-specific annotations, e.g. defining imprecise analogue signals.

- Specifications should be given in a natural, typical way for an analogue design engineer. This requires the description of vague specifications, e.g. Fuzzy Relation Memories defined with linguistic variables.

- Allow the designer to draw graphs of signals, to specify graphs of signals, and to model imprecise nonlinear systems using Fuzzy Relation Memories.

- Apply the most appropriate reasoning technique (e.g. qualitative analysis, numerical computation, fuzzy analysis) to the model to answer the designer's questions.

Most interesting for analogue design engineers is the possibility to express complex relationships, such as signals or temperature drift, etc. by signal graphs. These signal graphs are sketched on paper by the designer and their intention is to convey qualitative information about the shapes of curves and relative magnitudes rather than precise numerical values (see Fig. 11.1).

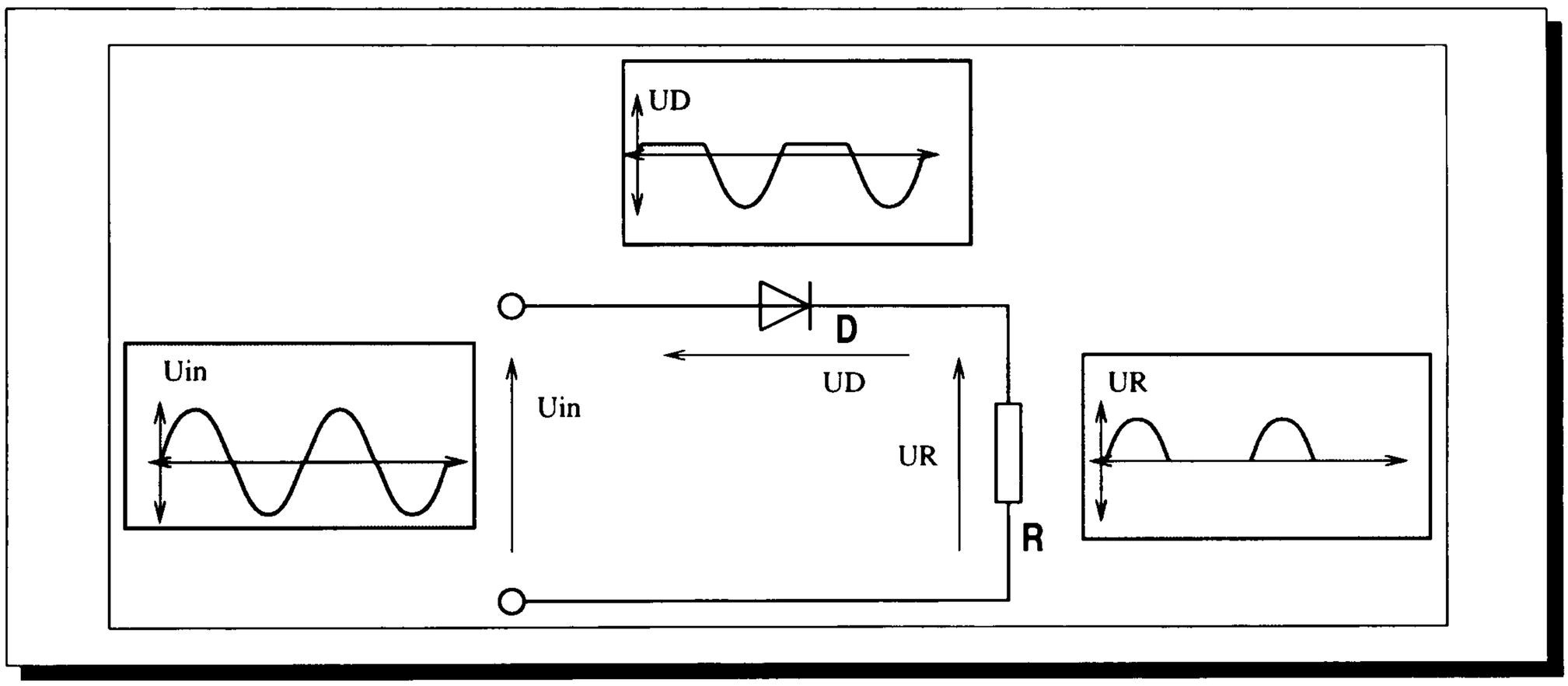


Figure 11.1: Circuit Paper Prototype Diode Circuit

Using signal graphs makes sense for analogue design engineers (or for other engineers and scientists) because they have powerful perceptual apparatus that helps them to decode signal graphs. Finding ways to make computers use signal graphs as easily and as flexibly as we do is a key problem in making software that can work better with people.

11.2 Fuzzy Relation Memories a Step Towards Graph Understanding

Experts frequently use graphical representations for presenting their information. Textbooks for mathematics, physics, or economics are hardly without any graphs and diagrams, and a blackboard bare of graph or diagram after a lecture in electronics is difficult to find. When asked to explain electronic circuits, experts in electronics found it extremely difficult to do so without referring to visual elements along with verbal explanations. In fact most circuit diagrams have visual elements of the electronic conditions included. The graphical representation serves as a place holder and to initiate reasoning, and finally holds, in effect, a summary of the expert's reasoning [Tabachneck et al., 1994].

One phenomenon that seems to be quite common is the use of mental imagery of design engineers (Larkin and Simon [Larkin and Simon, 1987]). With Fuzzy Relation Memories (Fuzzy Curves) design engineers are able to define imprecise specifications graphically.

Diagram understanding requires the ability to identify objects, determine the relevant features for a particular problem and map the graphical features to the domain. A signal graph is a specialized form of diagrammatic representation that does not involve object recognition. Line graphs, used for e.g. signals, show continuous changes in the domain of analogue circuit design.

A first attempt in reasoning about graphs was done by Yusuf Pisan [Pisan, 1995a],[Pisan, 1995b] for a system called SKETCHY. The system interprets drawings consisting of lines, regions, and curves. SKETCHY allows the processing of the arbitrary-shaped curves which are approximated by a large set of very short line segments. Comparative analysis of diagrams is an interesting feature for systems. SKETCHY can figure out how a change in one parameter of a diagram will affect another. This is the heart of sensitivity analysis, which is crucial in design

optimization. There have been efforts in graphical descriptions of digital circuits shown in [Smedley, 1996].

Fuzzy Curves especially transfered to the qualitative domain with linguistic variables could be a first step towards a system like SKETCHY for the analogue circuit design domain. Qualitative simulation has the potential of automatical interpretation of the simulation result. Only the values which are interesting for the designer are part of the qualitative domain. The result of a qualitative simulation requires much less effort to interpret.

11.3 Fuzzy Curve Simulation — Model Testing

The core of the **Circuit Paper Prototype Design Phase** is that the vaguely known specifications and imprecise circuit models can be defined by the designer at this stage of the design process. Especially the definition of imprecise signals and imprecise nonlinear modelling by Fuzzy Relation Memories is supported. The main purpose of this design phase is to develop a prototype model that satisfies the wanted behaviour. The model does not necessarily have to represent a real circuit but if so, it certainly increases the possibility of finding a real circuit for the prototype model. The simulation of the Circuit Paper Prototype by propagating Fuzzy Curves through the model, is achieved on the basis of qualitative fuzzy simulation as described in Part I.

To find out if the prototype model of the wanted circuit system under development is correct at the circuit paper prototype phase, the circuit is simulated with input data defined by the designer. The output of the simulation is compared with the specified wanted output data as visualized in Fig. 11.2.

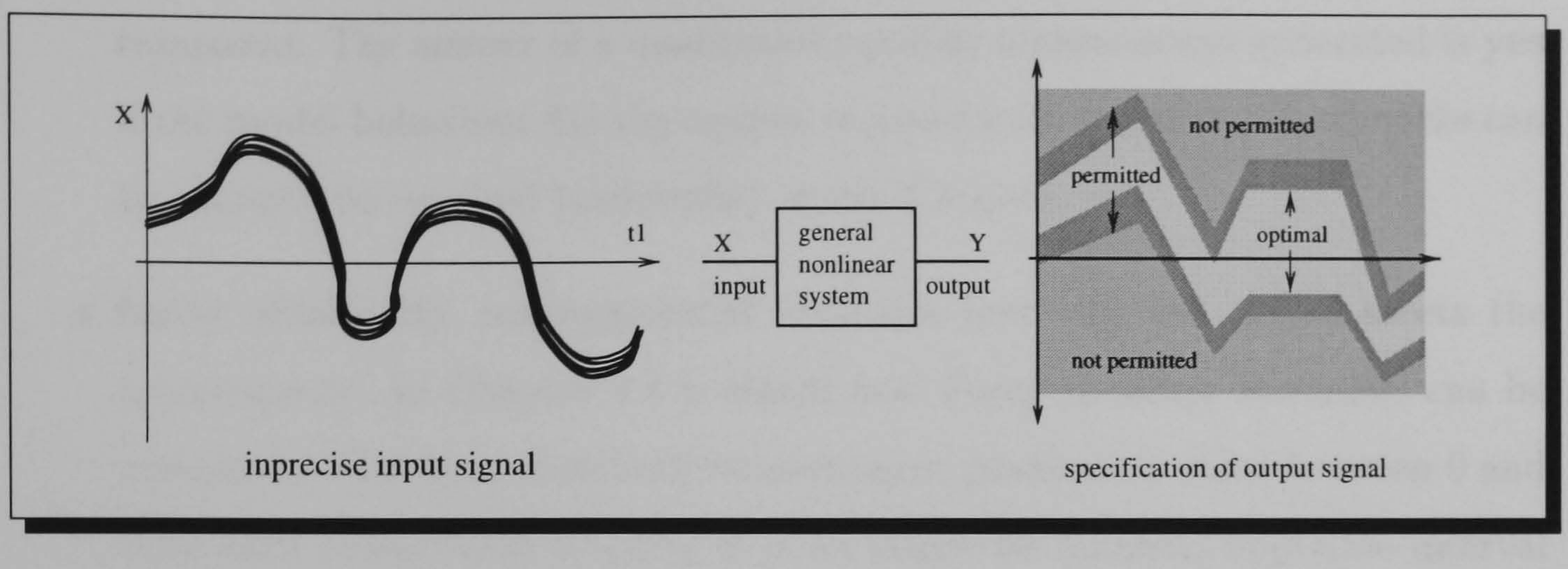


Figure 11.2: General Nonlinear System

Fig.11.2 shows a specification of a general nonlinear system with one imprecise input signal and one specified output signal.

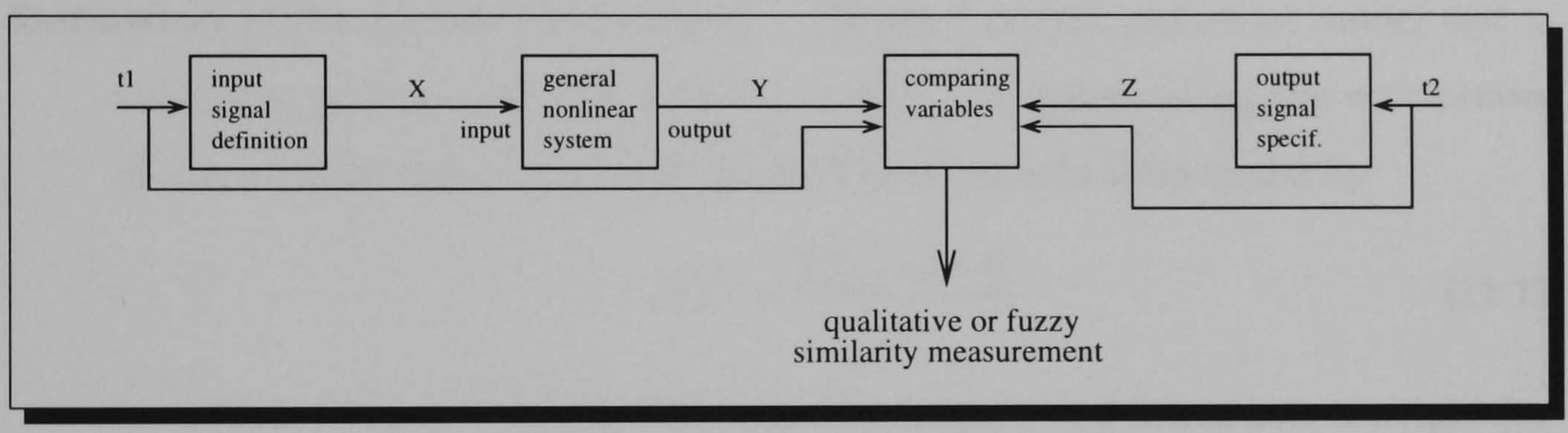


Figure 11.3: Fuzzy Similarity Measurement

The input signal is propagated through the nonlinear system and an output signal Y is generated. This signal is compared with the wanted output signal and a similarity measurement is determined (Fig. 11.3). There are two different measurements for comparing simulated output data with specified output data depending on the kind of simulation the design engineer performed:

- **qualitative equality measurement** indicates whether the model might meet the requirements or not. Since a qualitative simulation generates a sequence of landmarks the wanted specifications (requirements) are transfered to the qualitative domain and specification landmarks and simulated landmarks are

compared. The answer of a qualitative equality measurement generated is **yes** if the model behaviour fits the output requirements (simulated landmarks can be mapped to required landmarks) or **no** if it does not.

- **fuzzy similarity measurement** indicates how well the model meets the requirements. In Chapter 4.4 is stated how Fuzzy Relation Memories can be compared. The fuzzy similarity measurement generates a value between 0 and 1 for each requirement whether it is an imprecise number, imprecise interval or imprecise signal requirement.

To get an overall measurement indicating the correctness of the model combining all single similarity measurements the product is built.

Definition 11.74 (*Model Correctness*): Given a circuit prototype model and n similarity measurements S_i with $i = 1, 2, 3, ..., n$ representing the correctness of every single requirement the Model Correctness is determined by

$$MC = \frac{\sum_{i=1}^{n} w_i * S_i}{n} \tag{11.1}$$

w_i with $i = 1, 2, 3, ..., n$ is a value between 0 and 1 and introduces a weight for each similarity measurement S_i.

The result of each similarity measurement of the output data to the specification data is multiplied by a weight and added to a total model similarity called Model Correctness visualized in Fig. 11.4.

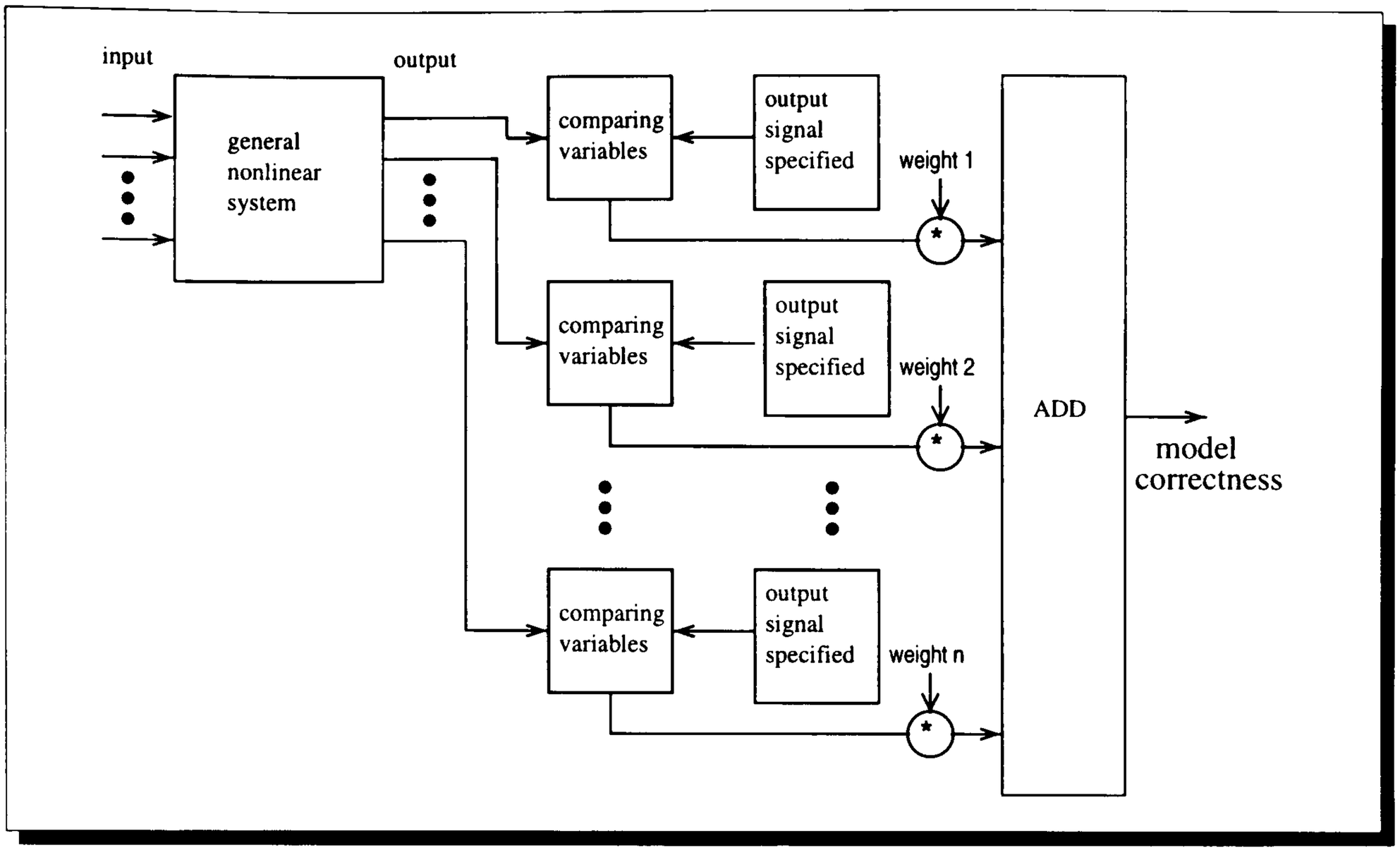


Figure 11.4: Circuit Paper Prototype Model Correctness

Whether a qualitative or a fuzzy simulation is performed depends entirely on the design engineer. The qualitative simulation is very often the first step analyzing the correctness of the circuit paper prototype model.

11.4 Prepare for Further Design

The prototype model must be prepared for the circuit configuration design phases (see Chapter 12). The basic functional blocks of the circuit prototype model have to be grouped. These groups should be represented by a real analogue circuit cell. The designer knows best, which basic functional blocks could be represented by real analogue circuit cells from the database. For example, with the prototype model shown in Section 11.5 the total model will probably be represented by just a single circuit cell.

11.5 Example: Amplifier Circuit

A designer wants to design an amplifier circuit (Fig. 11.5), that amplifies an input signal.

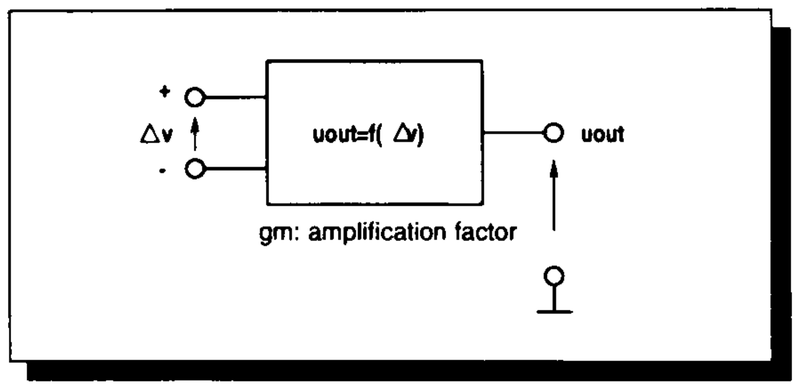


Figure 11.5: Block Model of an Amplifier Circuit

A possible rule based model of an amplifier circuit is shown below:

$$
\begin{array}{lll}
\text{IF} \quad (\Delta v < -\Delta \tilde{v_{max}}) & \text{THEN} & \tilde{uout} = -\tilde{uout}_{max} \\
\text{IF} \quad (\Delta v \geq -\Delta \tilde{v_{max}}) \text{ AND} & & \\
\quad\quad (\Delta v \leq \Delta \tilde{v_{max}}) & \text{THEN} & \tilde{uout} = \tilde{gm} * \Delta v \\
\text{IF} \quad (\Delta v > \Delta \tilde{v_{max}}) & \text{THEN} & \tilde{uout} = \tilde{uout}_{max}
\end{array}
$$

Fig. 11.6 shows the model as a Fuzzy Relation Memory with:

$$uout = f_1 = -uout_{max} \quad uout = f_2 = gm * \Delta v \quad uout = f_3 = uout_{max}$$

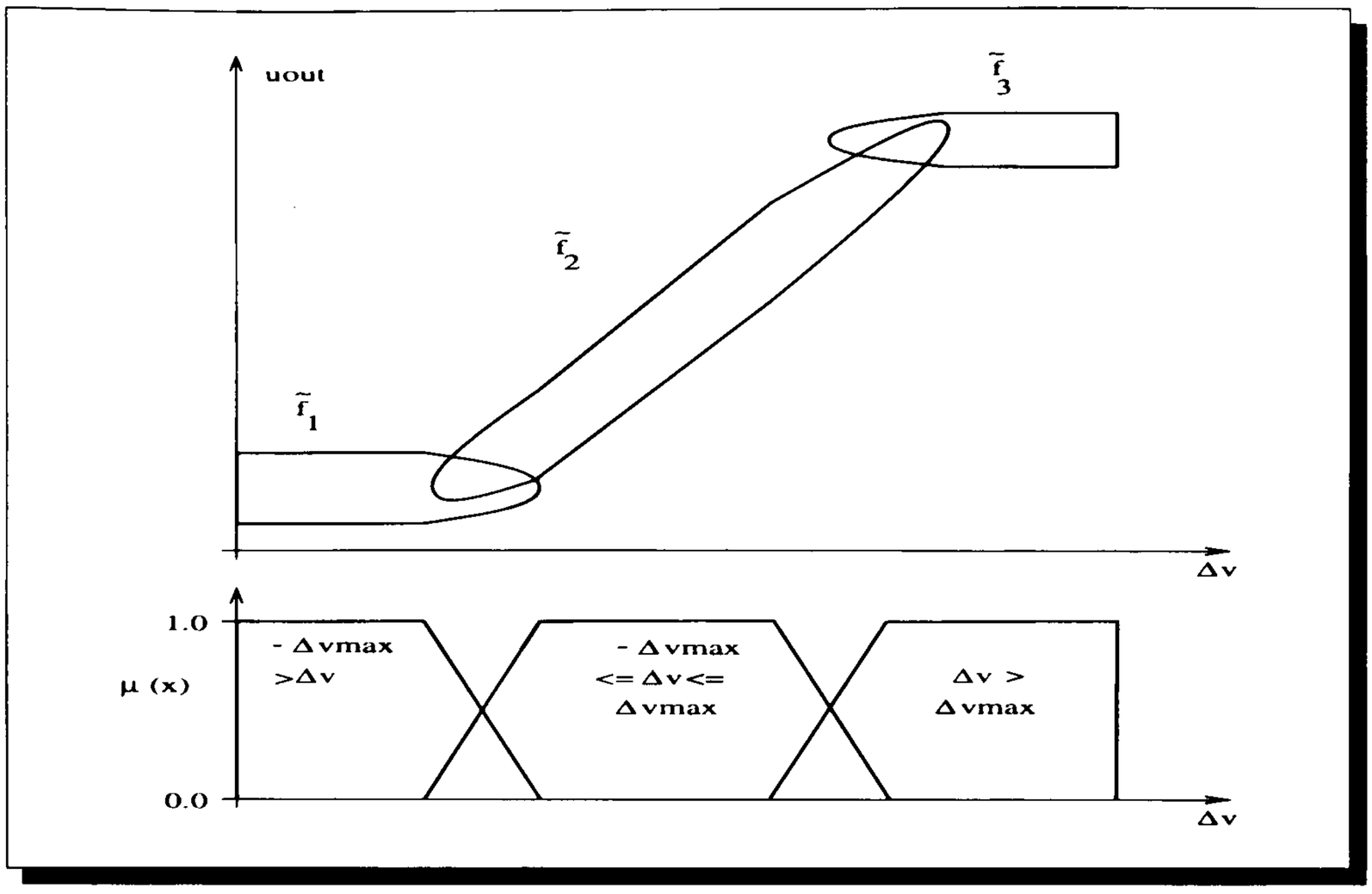


Figure 11.6: Fuzzy Relation Memory Model of an Amplifier Circuit

11.5.1 Qualitative Simulation of Amplifier Circuit Model

Suppose at first the designer wants to do a qualitative simulation and therefore defines a qualitative domain:

$$Q = \{-Vdd, -uout_{max}, -uout_1, -In_{max}, -\Delta v_{max}, -\tfrac{\Delta v_{max}}{2}, Zero,$$
$$\tfrac{\Delta v_{max}}{2}, \Delta v_{max}, In_{max}, uout_1, uout_{max}, +Vdd, gm\}$$

These given landmarks (symbols) are of significance for the designer. The symbols are defined by fuzzy landmarks (see Chapter 3) as following:

$$
\begin{aligned}
-Vdd &= \{-15.000, 0.100, 0.100\} \\
-uout_{max} &= \{-12.000, 0.100, 0.100\} \\
-uout_1 &= \{-8.000, 0.100, 0.100\} \\
-In_{max} &= \{-0.012, 0.001, 0.001\} \\
-\Delta v_{max} &= \{-0.006, 0.001, 0.001\} \\
-\frac{\Delta v_{max}}{2} &= \{-0.003, 0.001, 0.001\} \\
Zero &= \{0.000, 0.001, 0.001\} \\
\frac{\Delta v_{max}}{2} &= \{0.003, 0.001, 0.001\} \\
\Delta v_{max} &= \{0.006, 0.001, 0.001\} \\
In_{max} &= \{0.012, 0.001, 0.001\} \\
uout_1 &= \{8.000, 0.100, 0.100\} \\
uout_{max} &= \{12.000, 0.100, 0.100\} \\
Vdd &= \{15.000, 0.100, 0.100\} \\
gm &= \{2000.000, 200.000, 200.000\}
\end{aligned}
$$

Figure 11.7: Definitions of the Vague Landmarks

The intervals between the fuzzy landmarks are defined by symmetric partitioned fuzzy intervals. Fig. 11.8 shows part of the linguistic variable representing the qualitative domain.

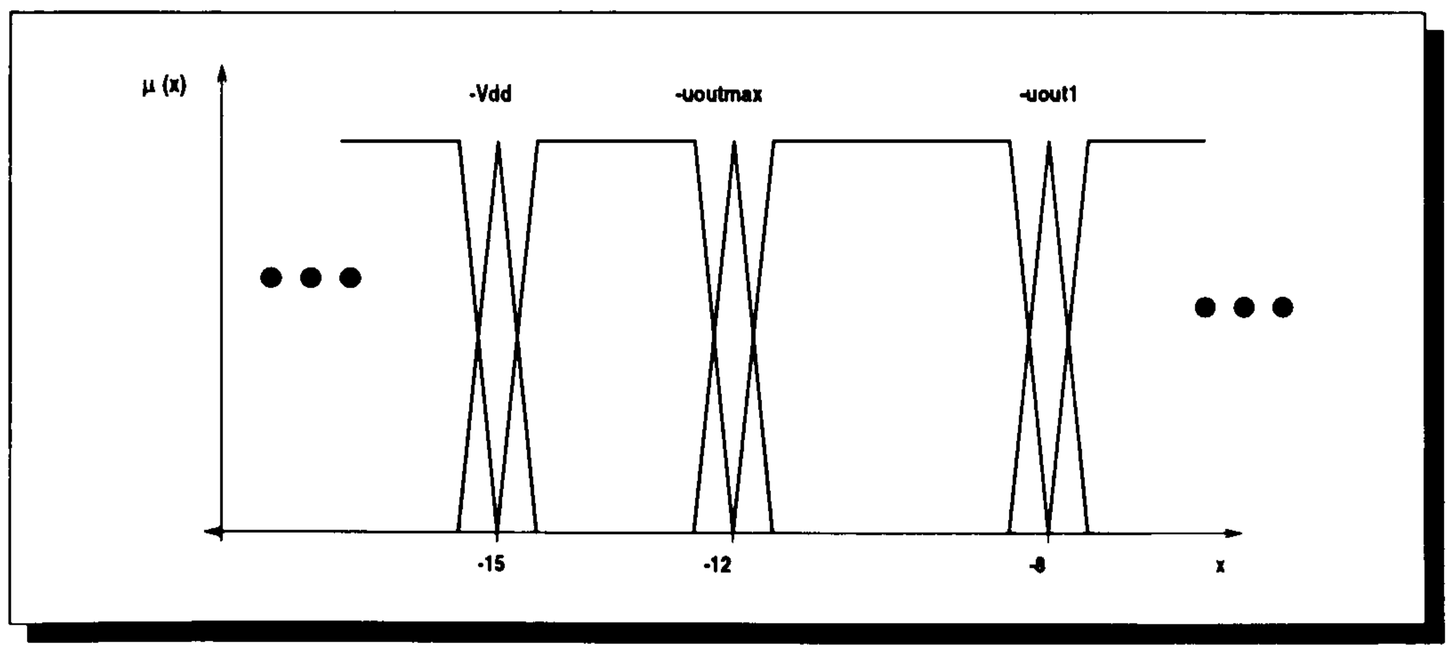


Figure 11.8: Linguistic Variable Defines Qualitative Domain (Partial View)

Having a real input signal (Fig. 11.9) as shown below:

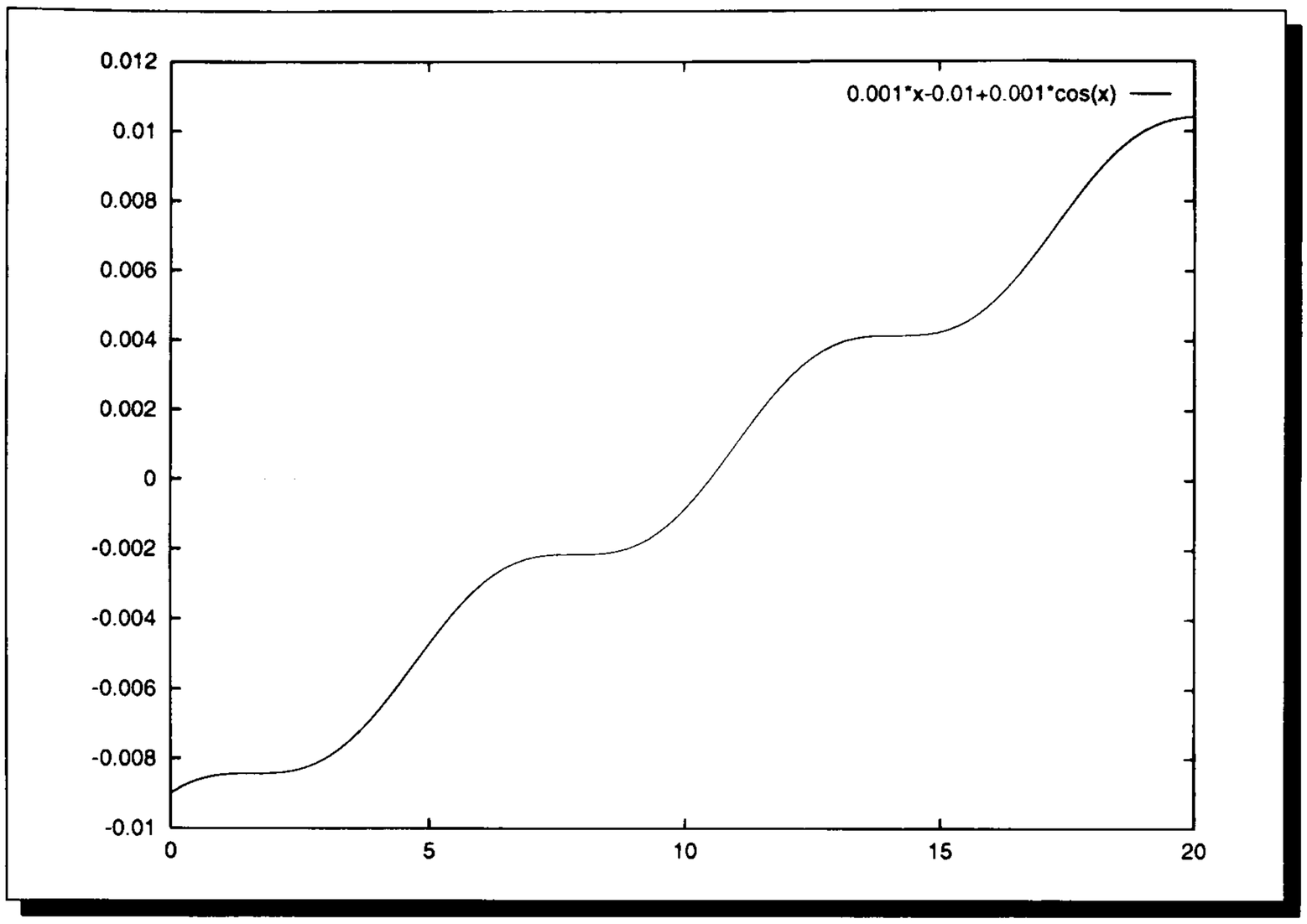


Figure 11.9: Real Input Signal to the Amplifier

Transferring the real input to a qualitative input function (Fig. 11.10) results in:

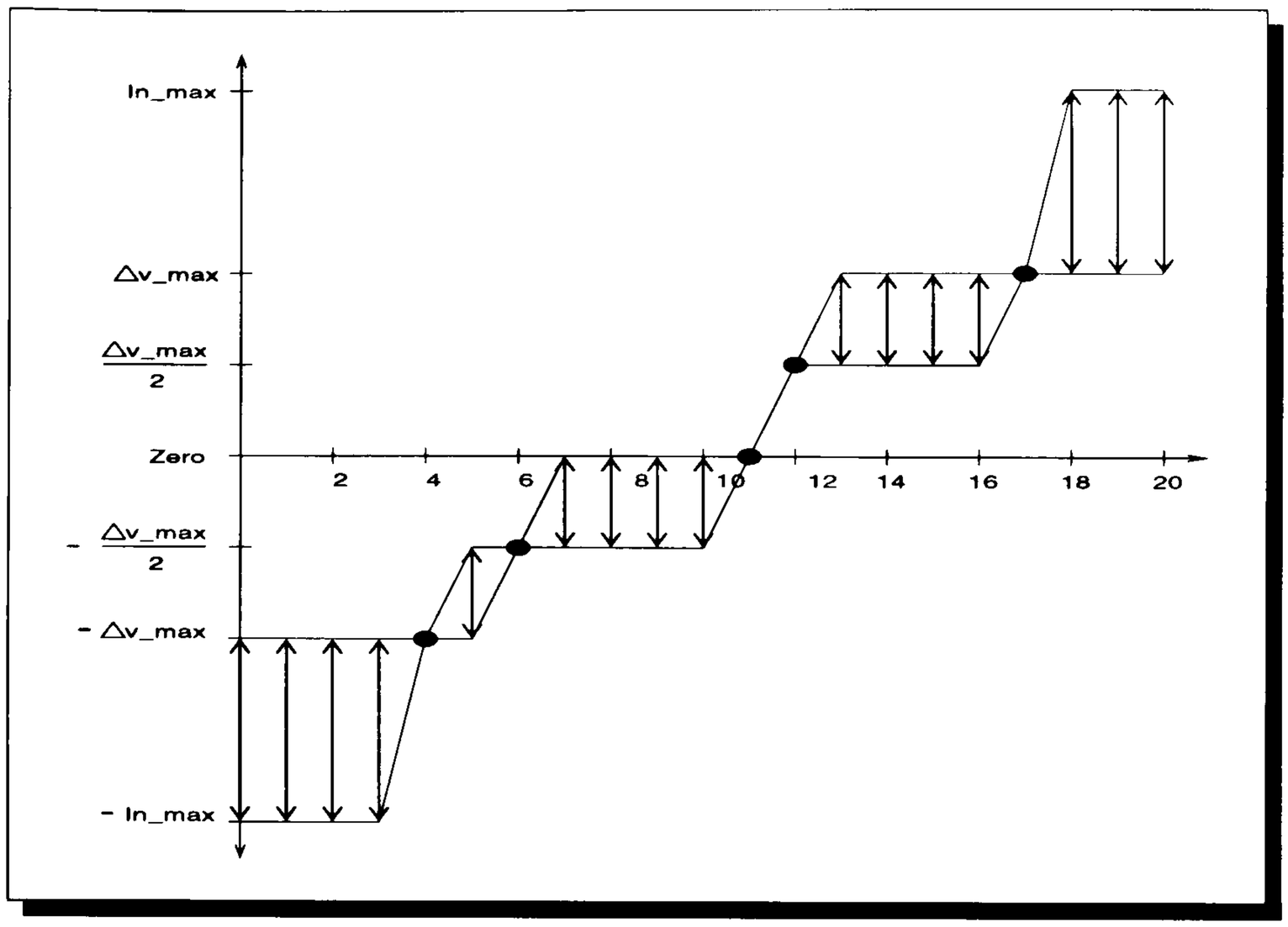
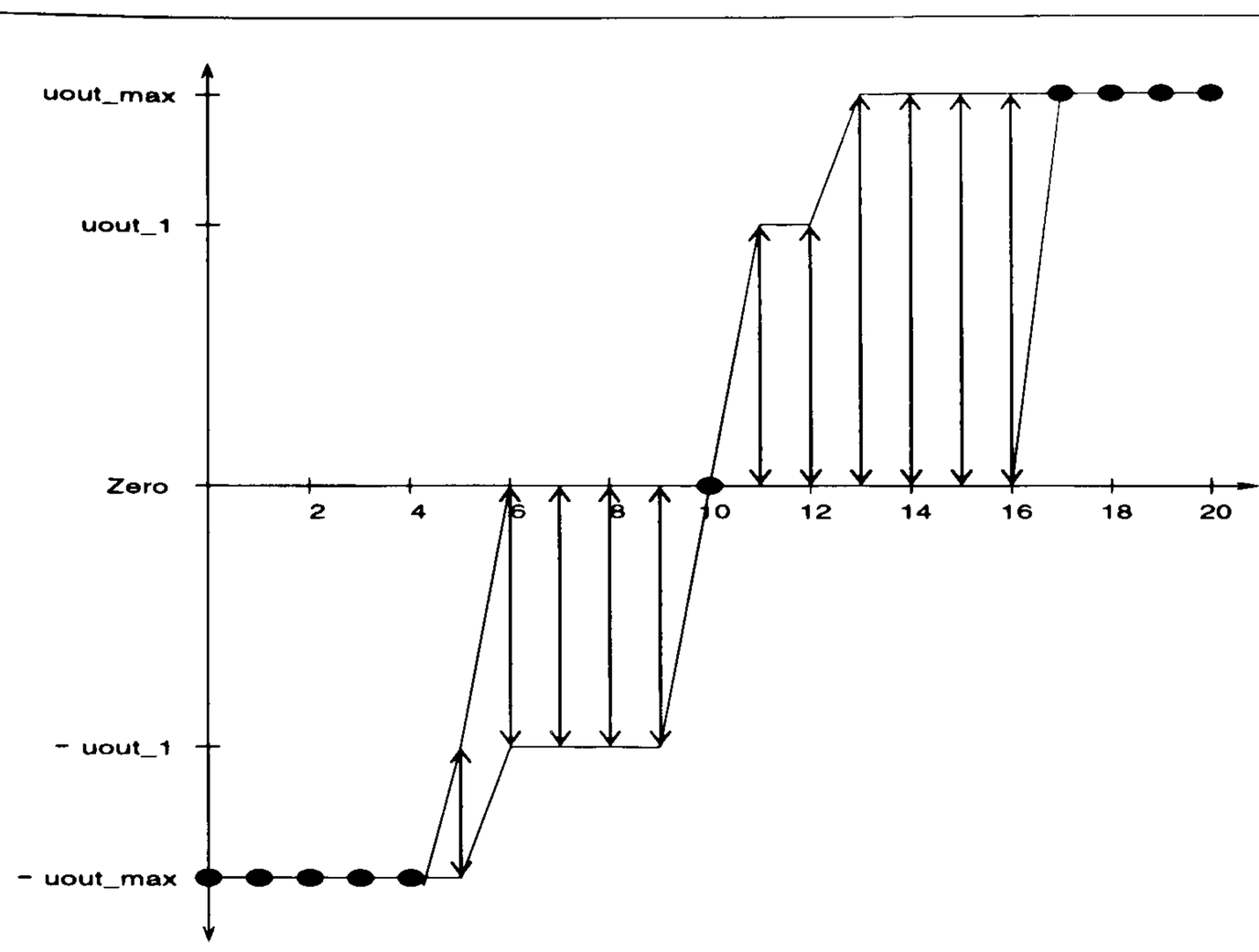


Figure 11.10: Qualitative Input Signal to the Amplifier Model

After a qualitative simulation we get the following result (Fig. 11.11) for the *uout*-signal:



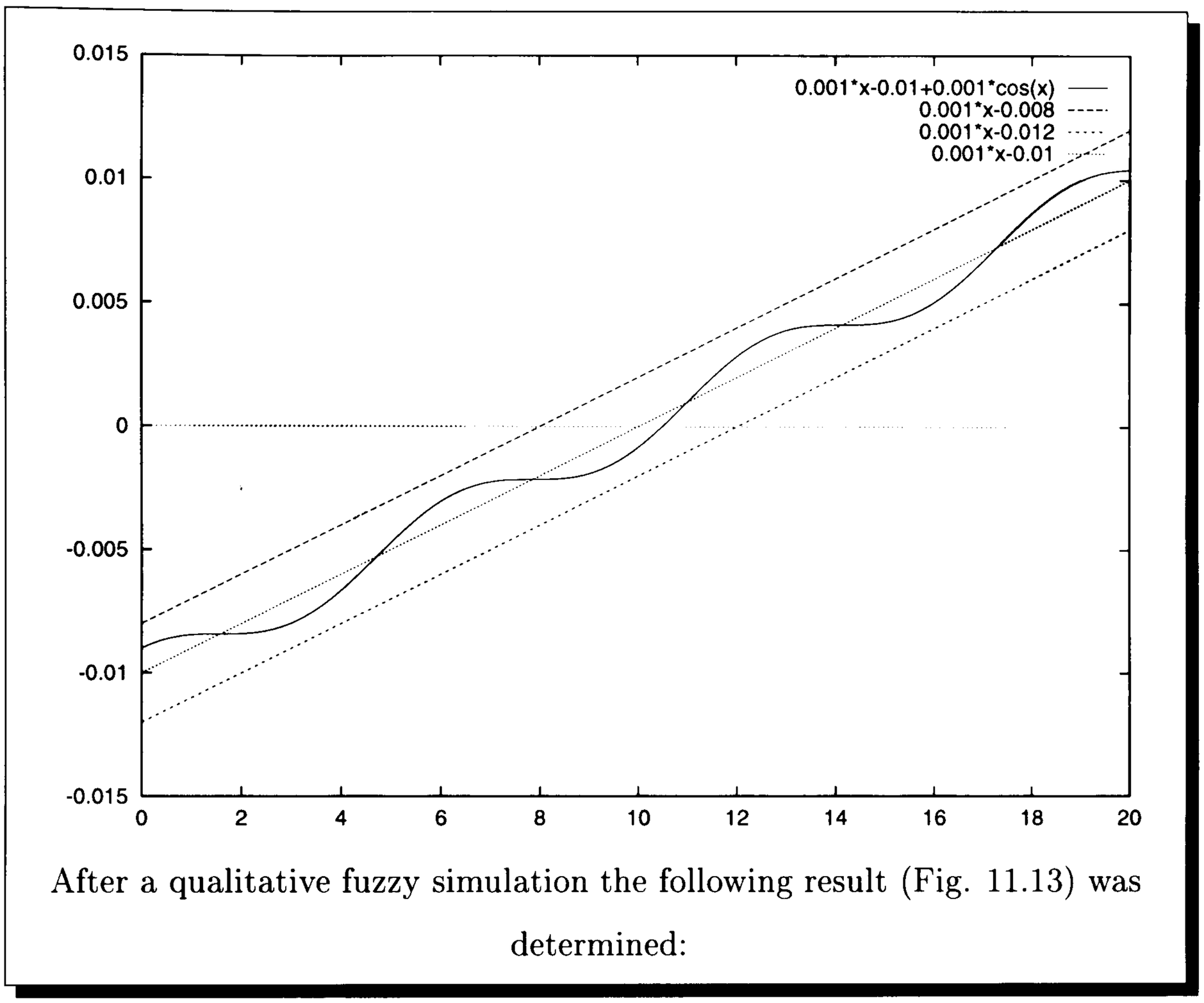
After a qualitative fuzzy simulation the following result (Fig. 11.13) was determined:

Figure 11.11: Qualitative Output Signal to the Amplifier Model

The Fig. 11.10 and Fig. 11.11 missed out some of the landmarks because the diagram axes are linear partitioned.

11.5.2 Fuzzy Simulation of Amplifier Circuit Model

Suppose the designer represents the real input (Fig. 11.10) by a fuzzy curve (Fig. 11.12).



After a qualitative fuzzy simulation the following result (Fig. 11.13) was determined:

Figure 11.12: Input Fuzzy Curve to the Amplifier Model

After a qualitative fuzzy simulation the following result (Fig. 11.13) was determined:

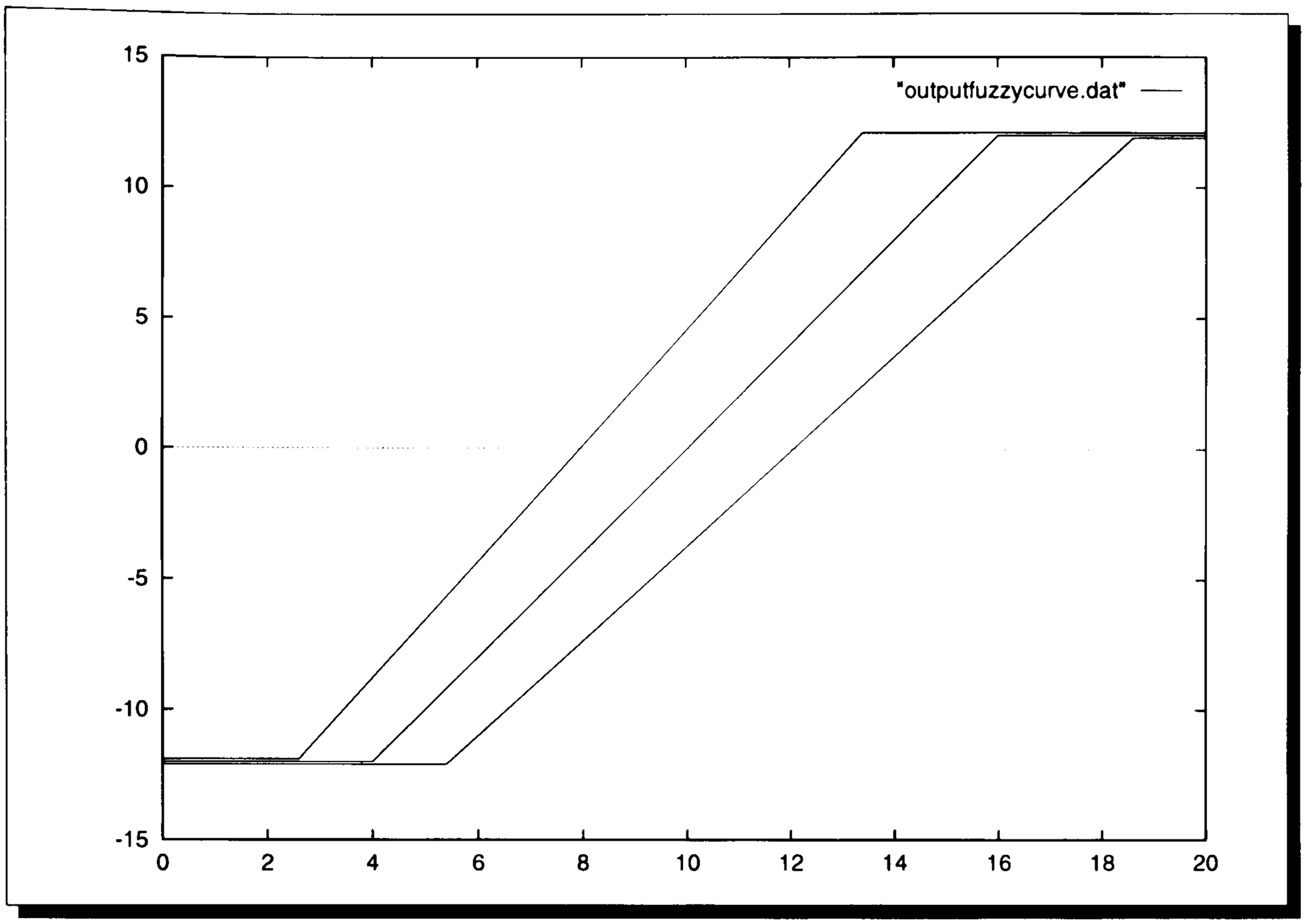


Figure 11.13: Output Fuzzy Curve of the Amplifier Model

Fig 11.13 shows three curves: one inner curve ($\alpha - cut - level = 1.0$) and two boundary curves ($\alpha - cut - level > 0.0$).

The inner curve shows the optimal simulated behaviour of the amplifier circuit model. The simulation using the optimal values of the input signal and the model parameter does not introduce any imprecision. The inner curve (Fig 11.13) of the model simulation shows a typical operational amplifier behaviour.

The two boundary curves represent the model simulation result at the $\alpha - cut - level > 0.0$. The result states the possible imprecision if the maximum imprecision of the input signal and model parameters is used to simulate the model. As illustrated in Fig 11.13 the two boundary curves build a hysteresis section when the circuit model amplifies. Beyond that the amplifier is in saturation and does not amplify.

The simulation result of the amplifier model describes a typical behavioural model for an operational amplifier circuit shown in Fig. 11.14.

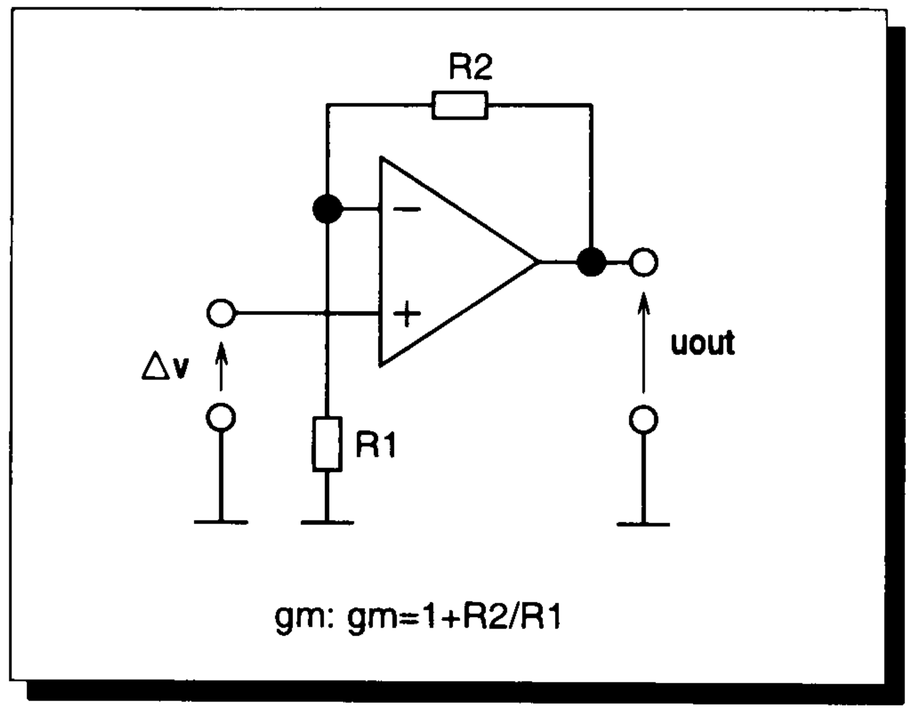


Figure 11.14: Operational Amplifier Circuit

11.6 Conclusion

Circuit paper prototype design supports the design of models at a very high-level of abstraction. At this stage of the design it is important for the designer to state the imprecision of the input/output data requirements and the vague modelling.

Using fuzzy curves improves the understanding and verification of the circuit model at an early stage of the design phase. Verifying the model using qualitative simulation indicates whether the model might meet the requirements and similarity measurement using fuzzy simulation determines how well the model meets the requirements.

The interpretation of the simulation results represented as qualitative or fuzzy values is correct if the same person who defined the specifications is interpreting the results. A drawback of this approach is that different user might interpret the same simulation results differently in respect of the imprecision of model parameters and specifications involved.

In general the qualitative fuzzy simulation results illustrate the stability and

robustness of a system model. The robustness of a system model is a statement about the susceptibility to disturbances. The imprecise a simulation output the less robust the model the more susceptible for disturbances.

Chapter 12

Qualitative and Fuzzy Configuration-Design

The synthesis problem is expressed as extracting the most suitable analogue circuit from an analogue circuit cell database. The circuit cell that meets the specifications best is selected from the database. This chapter describes the design phase: qualitative and fuzzy configuration-design of analogue circuits in more detail. The first section shows how the analogue circuit cells are described in the pre-simulated database. Followed by examples that are given for the new research methodologies; **qualitative configuration-design** and **fuzzy configuration-design** of analogue circuits.

12.1 Pre-Simulated Analogue Circuit Cell Database

The analogue circuits in a database of a company have been designed having its special purpose. There are circuits whose purpose are to amplify, to stabilize, to detect, to attenuate, etc. During the design process these circuit cells have been simulated using input data, e.g. step functions, sinusoidal functions, mixed DC/AC

functions, etc. After examination of the simulated output data and comparison with the desired behaviour the analogue circuit is appropriate and saved in the circuit cell database or redesigned. These circuit networks in the database are usually modeled by SPICE network descriptions ([Nagel, 1973]). SPICE is the most widely available computer program for the analysis of electronic networks [Grimbleby, 1990]. The input data that caused the desired behaviour for a particular circuit function is typically not saved.

This work proposes that a database of analogue circuits are stored with their input and output data determined analyzing at their intended working condition. Sets of input data and sets of output data represent the behaviour of the circuit and reflect the intended purpose, the function of the circuits. The input and output data sets are characteristic sets for the behaviour of the circuit. Looking for a suitable circuit cell means using the input data, simulate it with the model of the circuit paper prototype, and compare simulated output data with the output data of the database (see Chapter 11.3). If the simulated output data and the output data of the database are equal the model of the circuit paper prototype and the circuit cell of the database are corresponding. Correspondence is measured by Model Correctness (Chapter 11.3) defined differently in qualitative configuration-design (see Section 12.2) and fuzzy configuration-design (see Section 12.3).

It is important to note that the circuit paper prototype model is independent of the model of a circuit of the circuit cell library. Using the same input data for simulating the circuit paper prototype model and the circuit cell, the response behaviour of the two are compared. The circuit of the database itself, is treated as a black-box with input and output terminals, as shown in Fig. 12.1.

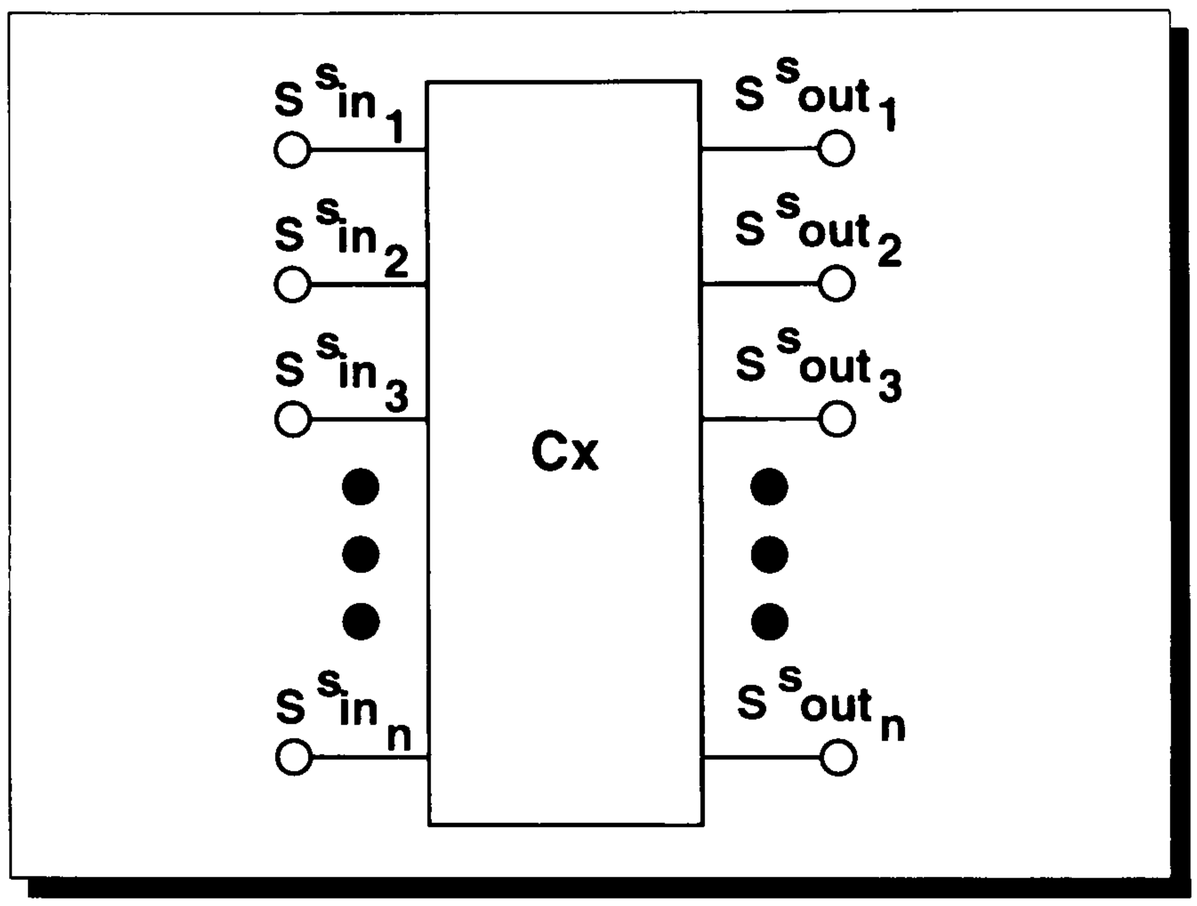


Figure 12.1: Generalized Black-Box Description of a Circuit in the Database

Of no interest are power supplies terminals and ground terminals, if they contribute nothing to the behaviour of the circuit. The description of the terminals is defined as follows:

$S^s in_n$ is an input; whereas	S:	is either a voltage (V) or a current (I) signal
	s:	is either a time (t) or a frequency (f) signal
	n:	index which distinguishes the different input
$S^s out_n$ is an output; whereas	S:	is either a voltage (V) or a current (I) signal
	s:	is either a time (t) or a frequency (f) signal
	n:	index which distinguishes the different output

The input and output data is stored in form of x and y data. The following example shows how a simple emitter-amplifier with emitter-resistor (Fig. 12.2) is stored in the database.

12.1.1 Storage Example of a Simple Amplifier Circuit in the Database

Fig. 12.2 shows a simple emitter-amplifier with emitter-resistor *Re*.

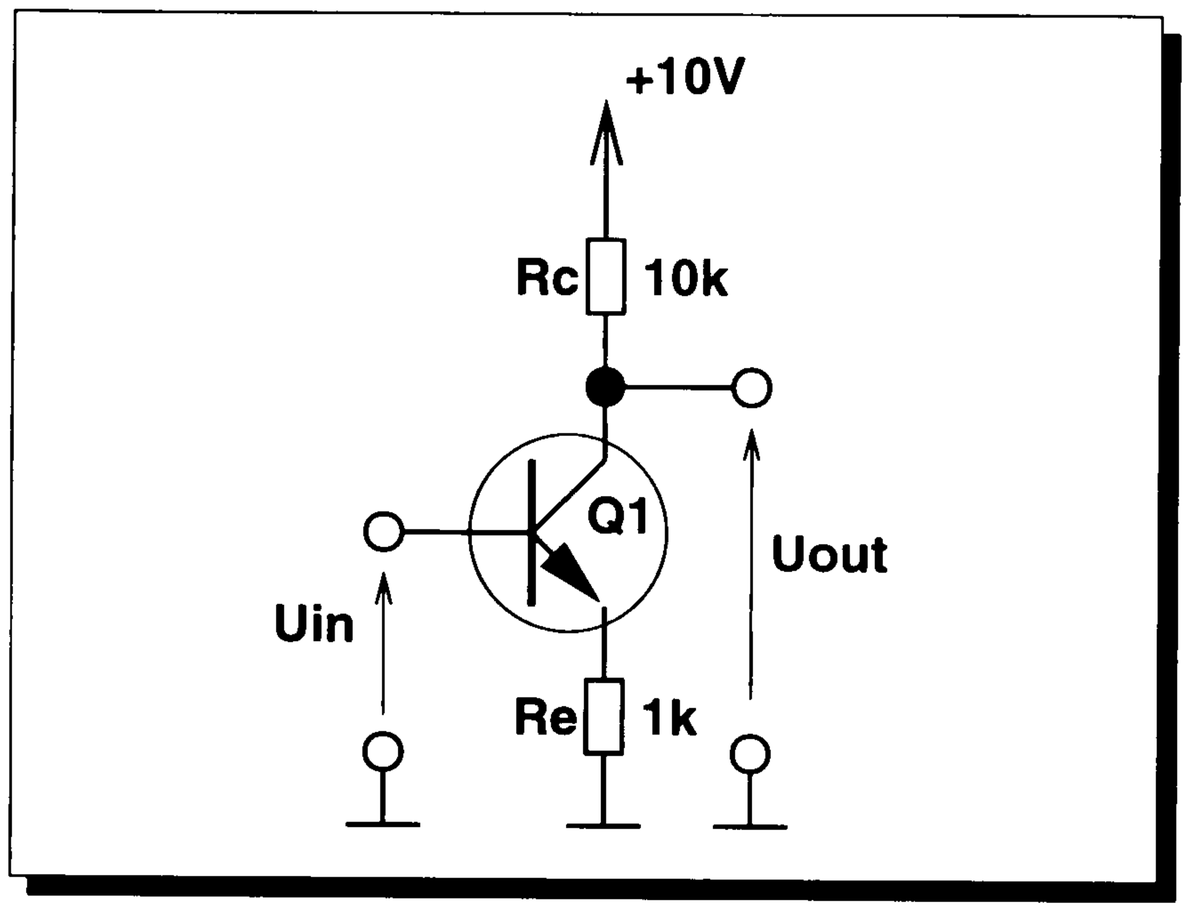


Figure 12.2: Emitter-Amplifier with Emitter-Resistor

The analogue amplifier circuit (Fig. 12.2) has been simulated using SPICE (Version: 2g.6) with the circuit description (see Table 12.1). The input signal at *Uin* is a sinusoidal function $0.2V * sin(2 * pi * 1k)$ with a constant part 1.13682V as stated in the SPICE network description file as `Vin 1 0 sin 1.13682 .2 1k`, see Table 12.1.

```
Emitter-Amplifier with Emitter-Resistor
Vcc 3 0 10
Rc 3 2 10k
Re 4 0 1k
Q1 2 1 4 tr1
.model tr1 npn is=10f vaf=100
Vin 1 0 sin 1.13682 .2 1k
.tran 10u 1m
.plot tran V(1,0)
.plot tran V(2,0)
.end
```

Table 12.1: SPICE Network Description File

After simulation the input signal and the output signal are shown in Fig. 12.3 and Fig. 12.4.

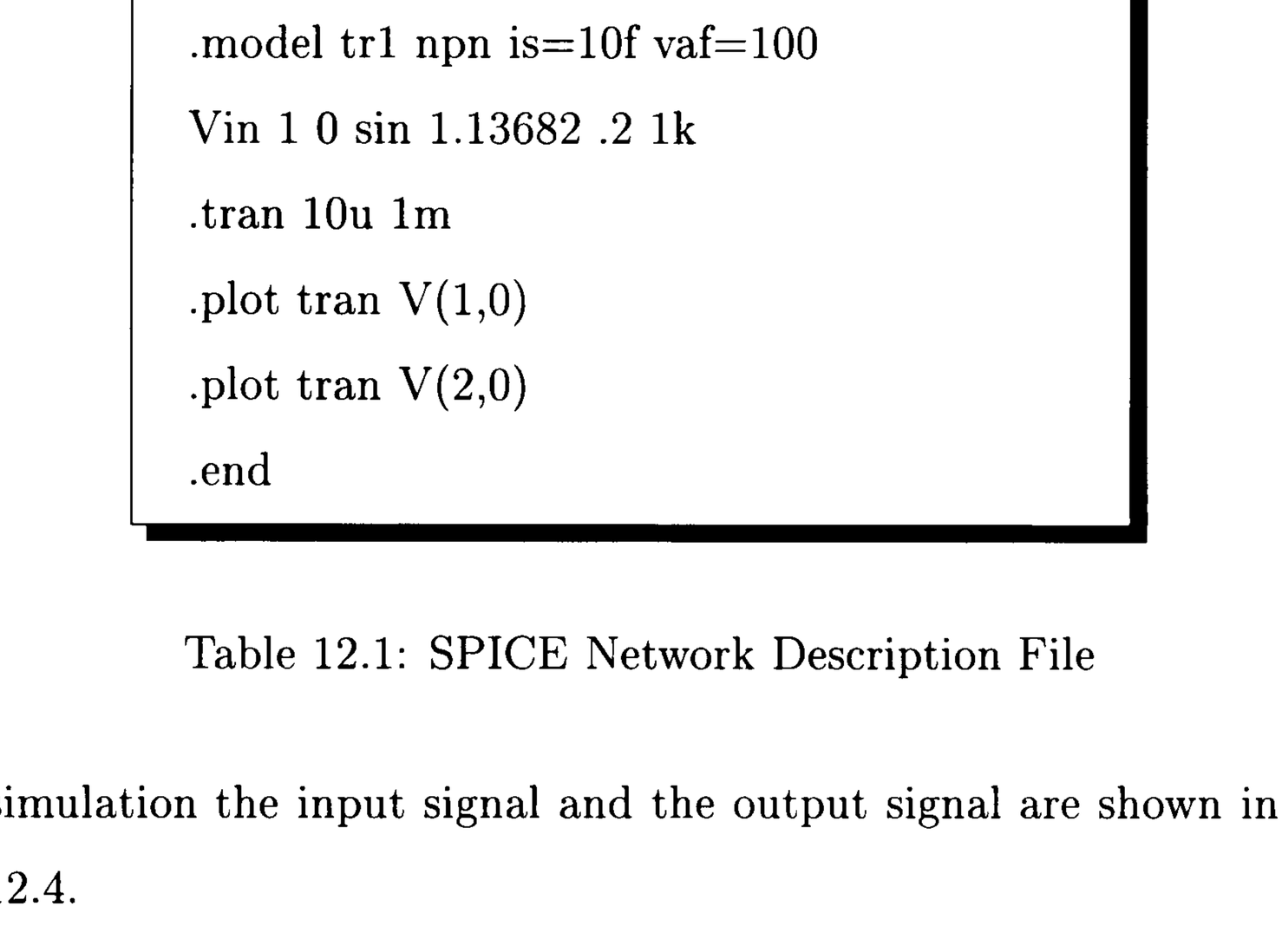


Figure 12.3: Input Signal for Simple Emitter Amplifier Circuit

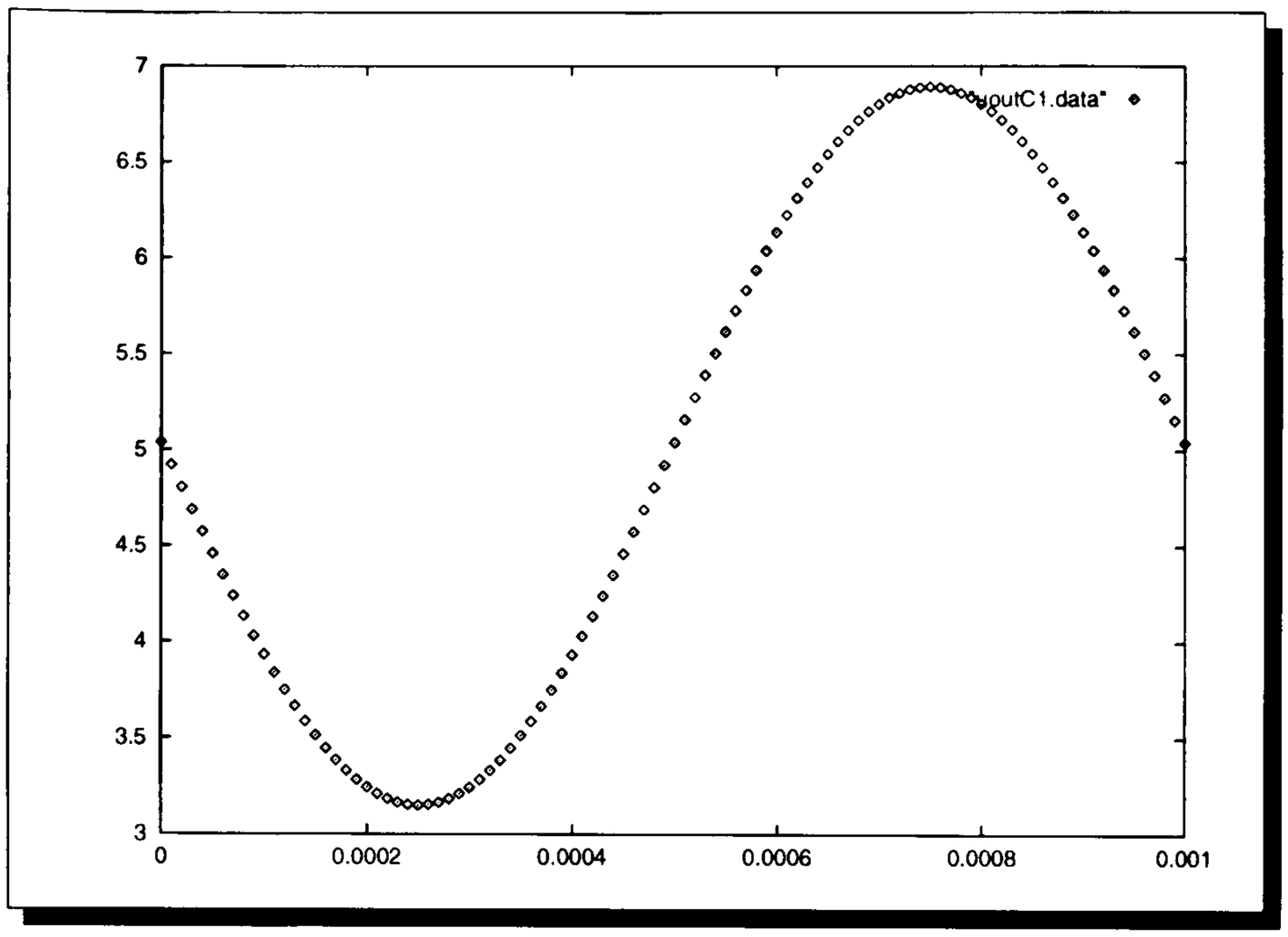
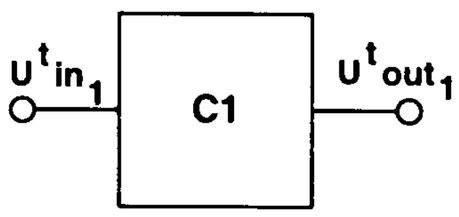


Figure 12.4: Output Signal for Simple Emitter Amplifier Circuit

The black-box description and its input and output data for the analogue amplifier circuit (Fig. 12.2, called C1), is shown in Fig. 12.5.

input data:

time t	voltage v
0.000E+00	1.137E+00
1.000E-05	1.149E+00
2.000E-05	1.162E+00
3.000E-05	1.174E+00
4.000E-05	1.186E+00
5.000E-05	1.199E+00
6.000E-05	1.210E+00
7.000E-05	1.222E+00
8.000E-05	1.233E+00
9.000E-05	1.244E+00
.	.
.	.
.	.
9.800E-04	1.112E+00
9.900E-04	1.124E+00
1.000E-03	1.137E+00



output data:

time t	voltage v
0.000E+00	5.039E+00
1.000E-05	4.921E+00
2.000E-05	4.804E+00
3.000E-05	4.687E+00
4.000E-05	4.572E+00
5.000E-05	4.458E+00
6.000E-05	4.347E+00
7.000E-05	4.238E+00
8.000E-05	4.132E+00
9.000E-05	4.030E+00
.	.
.	.
.	.
9.800E-04	5.274E+00
9.900E-04	5.157E+00
1.000E-03	5.039E+00

Figure 12.5: Black-Box Description and Input/Output Signal for Circuit C1

For more examples of implemented analogue circuit cells in the database see Appendix C.

12.2 Qualitative Configuration-Design

For qualitative configuration-design, as defined in Section 9.4.3, the designer has to define a qualitative domain. The qualitative domain consists of landmarks which

are quantities of interest for the designer. The designer uses intuition to choose the landmarks. Internally these landmarks are mapped to fuzzy values called fuzzy landmarks (see Chapter 3.5). The defined model at the circuit paper prototype phase is used to simulate qualitatively. After a qualitative simulation the output is compared qualitatively with the given output data of the circuit in the database. Is it equal to the results of the qualitative simulation, the circuit from the database might be suitable for further investigation at the next design phase; the fuzzy configuration-design.

Transferring the data into qualitative values given real values is mapping the real numbers to the qualitative domain. There are two major types of data:

- **Constant Data**: Data that is constant at any time of the simulation modeled by fuzzy landmarks and fuzzy qualitative intervals (see Chapter 3.5).
- **Dynamic Data**: Data that changes during simulation. There are time-dependent data and frequency-dependent data both modeled by Fuzzy Relation Memories (see Chapter 4.4).

12.2.1 Similarity of Simulated Qualitative Data and Database Data — Characterizing Analogue Circuits

As discussed in the Section "Historical Survey of Qualitative Reasoning" 2.1 the qualitative simulation might generate several results, the complete behaviour of the model. Since the landmarks are mapped to fuzzy values a defuzzification step could be used to filter out some of the simulation results. But at this level of the design process this is not performed. If one of the output data sets generated by the qualitative simulation is equal to the transfered output data the search for a possible circuit cell of the database has been successful.

Having completed this design stage the designer has several possible circuit structures which might meet the specifications. The qualitative simulation generates all

possible behaviour of the model. Thus circuits are extracted which might be not suitable at all. These are ruled out by the a more detailed specification and a more detailed simulation at the next stage: The Fuzzy Configuration-Design.

12.3 Fuzzy Configuration-Design of Analogue Circuits

To find out the most promising circuit structure the model of the circuit is simulated with the input data of the database and compared with the output data of the database. Only analogue circuit cells of the database which passed the qualitative configuration-design are used. Different to the qualitative configuration-design the data of the database do not have to be transfered, they are known exactly and therefore represented by real numbers. Only the imprecision of the model parameters are of significance to the fuzzy simulation result. Unlike to the qualitative simulation the fuzzy simulation has only one simulation result.

12.3.1 Analogue Circuit Similarity Measurement – Ordering of Found Analogue Circuits of the Database

The similarity of the simulated quantities to the specified quantities judges which of the given analogue circuits of the database meets the specifications best. Besides of the real and fuzzy values, the fuzzy curves are most interesting. How they are compared is defined in section 5.3: "Similarity Measurement of Fuzzy Curves".

Having systems which have more than one output the ordering of the found analogue circuits depend on whether the specifications of the output data are equally important. If not, the output data has to be weighted. The result of each similarity measurement of the output data to the specification data is multiplied by a weight and added to a total circuit similarity as defined as the model correctness definition in Chapter 11.

12.4 Configuration-Design Example: Amplifier

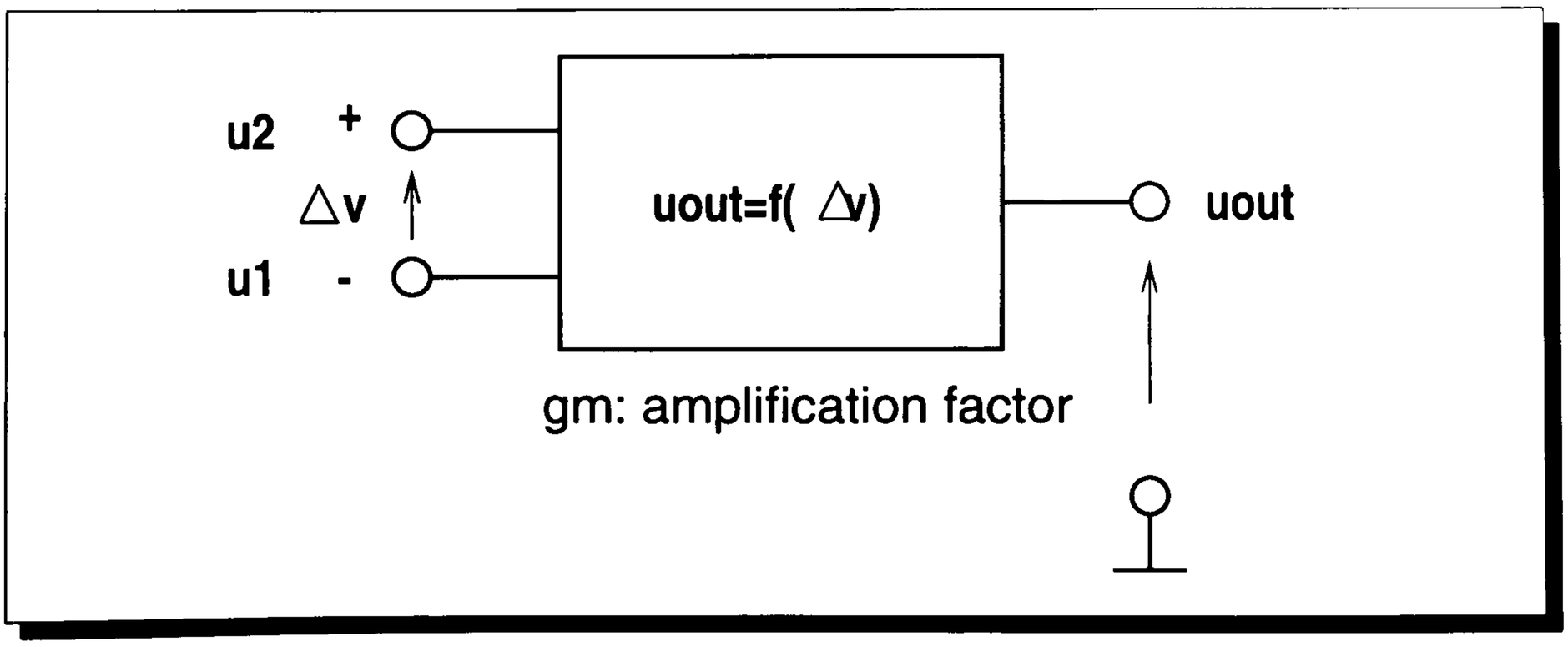
Relevant for qualitative configuration-design are the real input and output signals of the pre-simulated database transformed to qualitative signals using the model developed during the circuit paper prototype design phase of Chapter 11.5. The database data is only transformed to qualitative values to help the design engineer to interpret the data because the data is described entirely in the qualitative domain space the designer specified. For simulation the real data from the database is used and only the parameters of the prototype model is qualitatively described.

12.4.1 Amplifier Circuit Paper Prototype Model

Amplifier Functional Model

IF	$((v_2 - v_1) < -\Delta v_{max})$	THEN	$uout = -uout_{max}$
IF	$((v_2 - v_1) \geq -\Delta v_{max})$AND		
	$((v_2 - v_1) \leq \Delta v_{max})$	THEN	$uout = gm * (v_2 - v_1)$
IF	$((v_2 - v_1) > \Delta v_{max})$	THEN	$uout = uout_{max}$

Amplifier Black-Box Model



12.4.2 Qualitative Configuration-Design

First of all the amplifier black-box model is used to check the interface. Circuits from the circuit cell database (Appendix: C) are used. Only the following 3 circuits match the interface:

`C2: differential amplifier`

`C5: operational amplifier`

`C7: analog multiplier`

The `analog multiplier` (circuit C7) is probably not suitable to amplify voltage differences. Nevertheless the interface of this circuit fits the requirements.

Second the input and output signals of the pre-simulated circuits are used and transfered to the qualitative domain. Suppose for the designer the following fuzzy landmarks of interest:

$$
\begin{aligned}
-uout_{max} &= (-12.000, 0.1, 0.1) \\
-In_{max} &= (-0.012, 0.001, 0.001) \\
-\Delta v_{max} &= (-0.006, 0.001, 0.001) \\
-\frac{\Delta v_{max}}{2} &= (-0.003, 0.001, 0.001) \\
Zero &= (0.000, 0.001, 0.001) \\
\frac{\Delta v_{max}}{2} &= (0.003, 0.001, 0.001) \\
\Delta v_{max} &= (0.006, 0.001, 0.001) \\
In_{max} &= (0.012, 0.001, 0.001) \\
uout_{max} &= (12.000, 0.1, 0.1) \\
gm &= (2000, 10, 10)
\end{aligned}
$$

After a qualitative simulation with the input data sets of the 3 circuit cells the following results are generated and illustrated in Fig. 12.6 for $C2$, Fig. 12.7 for $C5$, Fig. 12.8 for $C7$ (output A), and Fig. 12.9 for $C7$ (output B). All figures show two solid lines an one dashed line. The solid lines represent the lower and upper boundary of the qualitative simulation result. The dashed line is stored in the database and has been generated using SPICE [Nagel, 1975].

For selecting only these circuits where the prototype model and the SPICE model of the database have the same behaviour, all values of the output data generated by SPICE must lay inside the qualitative simulation of the prototype model. Only circuit C5, as shown in Fig. 12.7, is a possible candidate for further processing. This means the prototype model and the SPICE model of circuit C5 correspond.

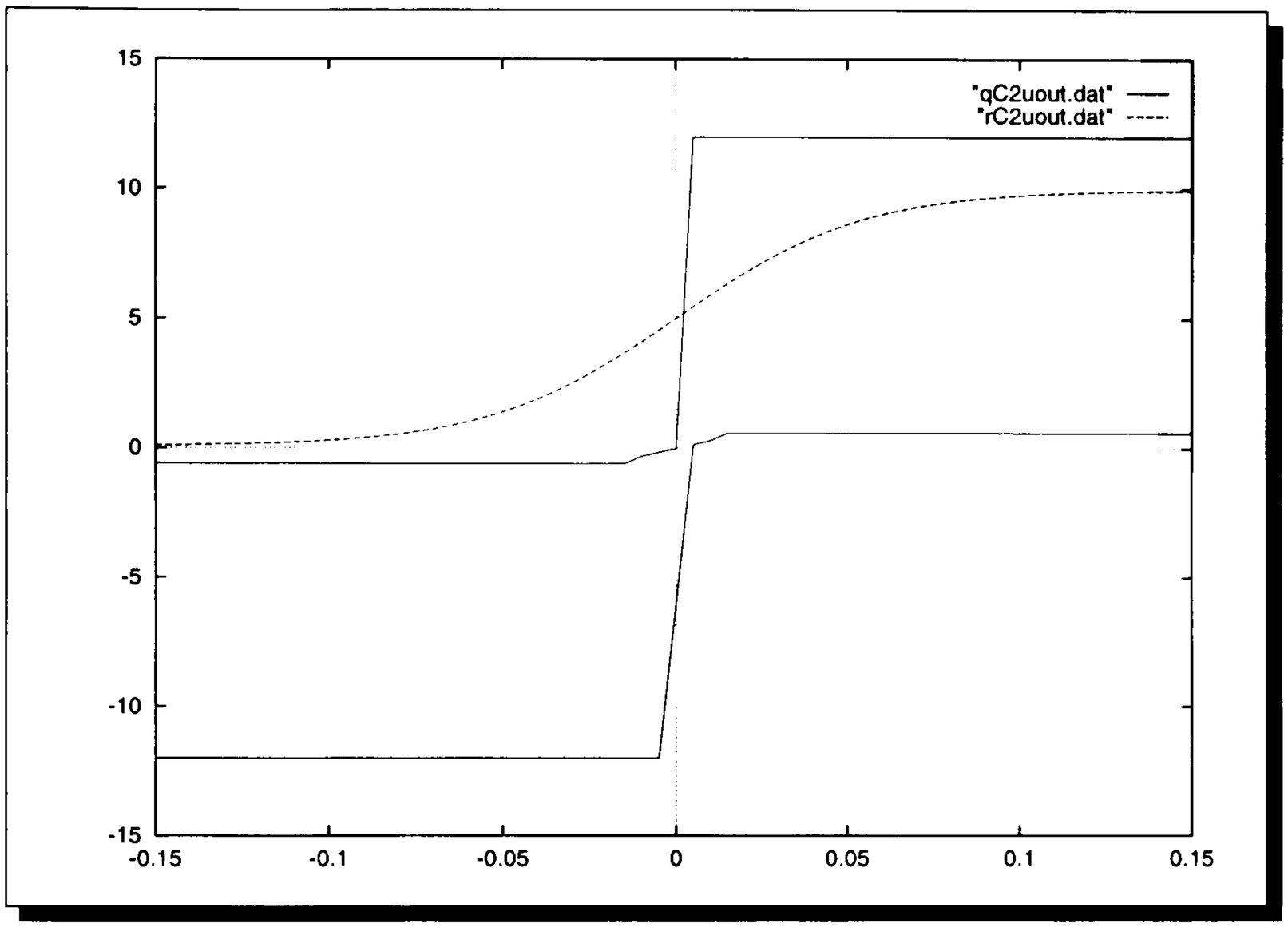


Figure 12.6: Result of Qualitative Simulation of C2

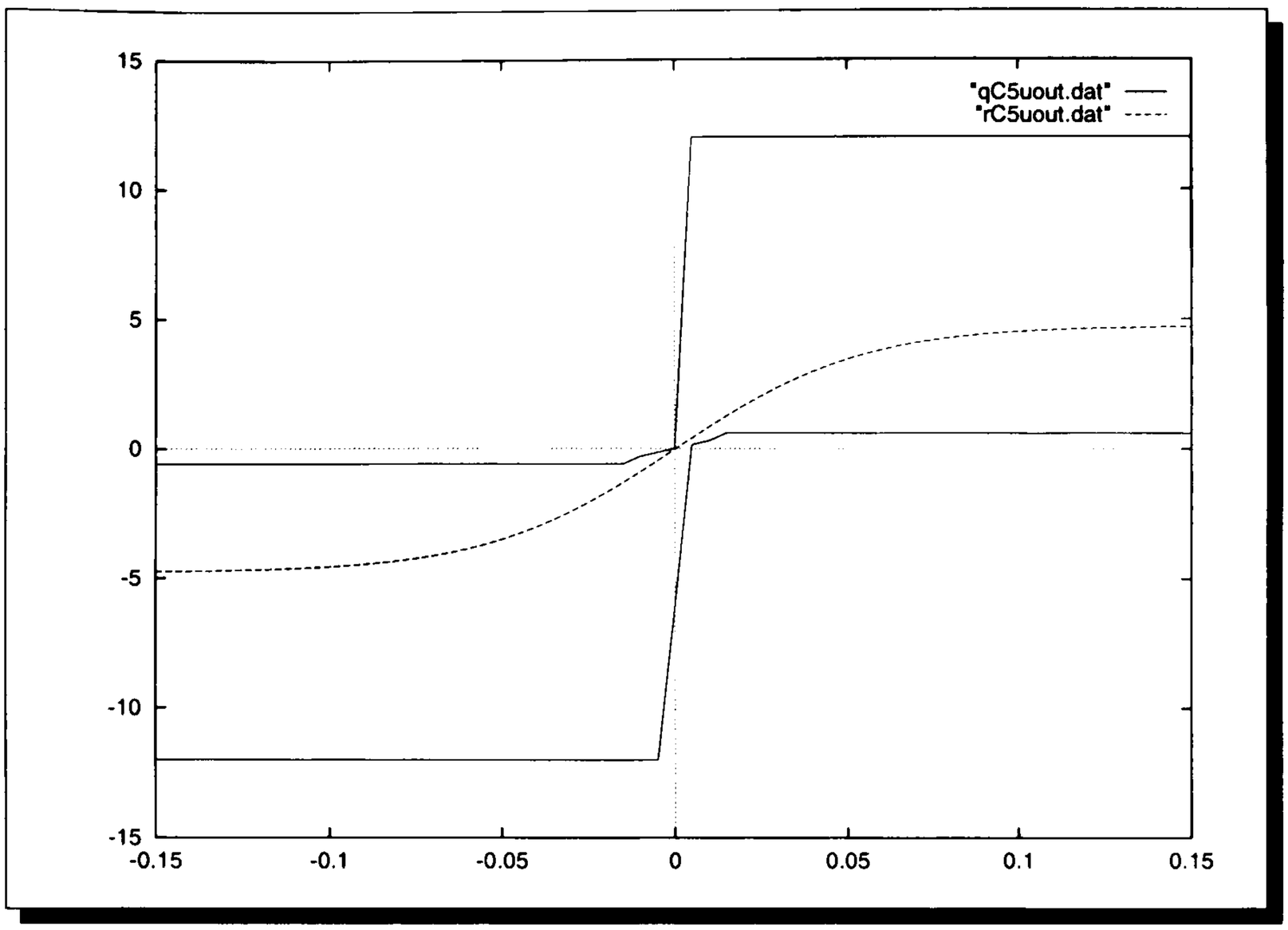


Figure 12.7: Result of Qualitative Simulation of C5

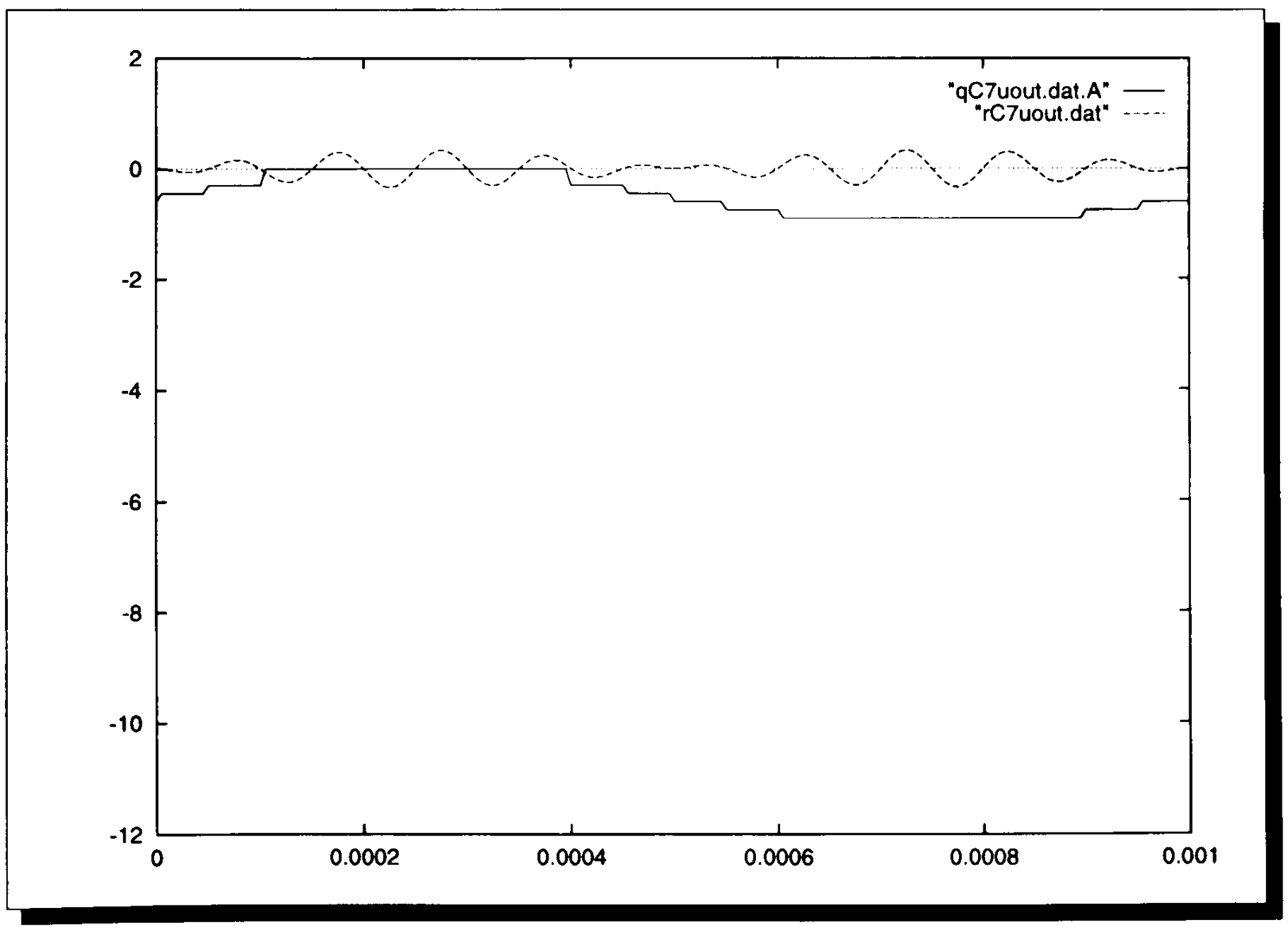


Figure 12.8: Result of Qualitative Simulation of C7: Output A

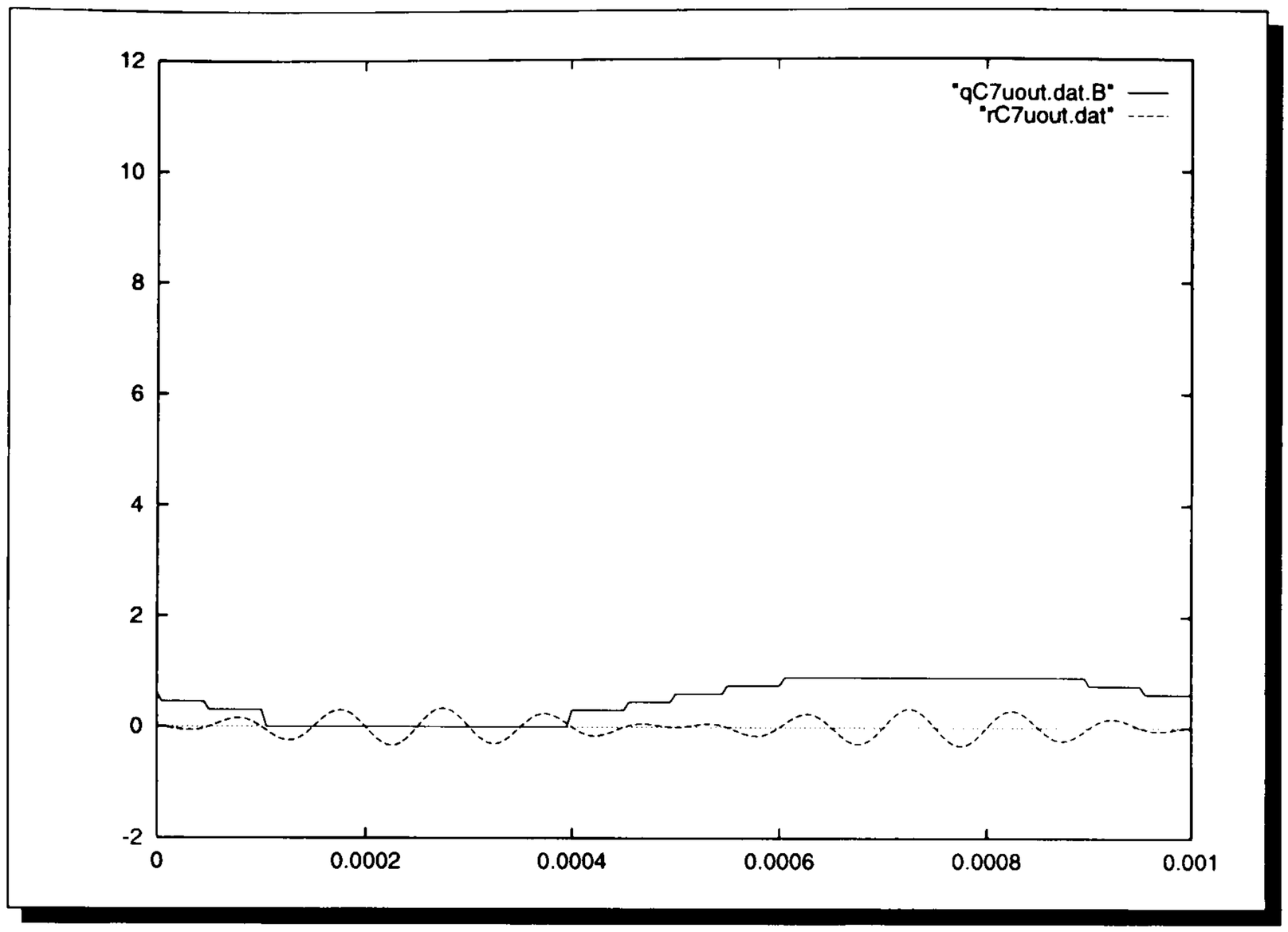


Figure 12.9: Result of Qualitative Simulation of C7: Output B

12.4.3 Fuzzy Configuration-Design

Suppose for the C5 circuit there are two slightly different circuits (A and B) stored in the database. To select the most promising circuit the prototype model is simulated at $\alpha - cut - level = 1.0$ and $\alpha - cut - level > 0.0$ generating the following simulation result (see Fig. 12.10).

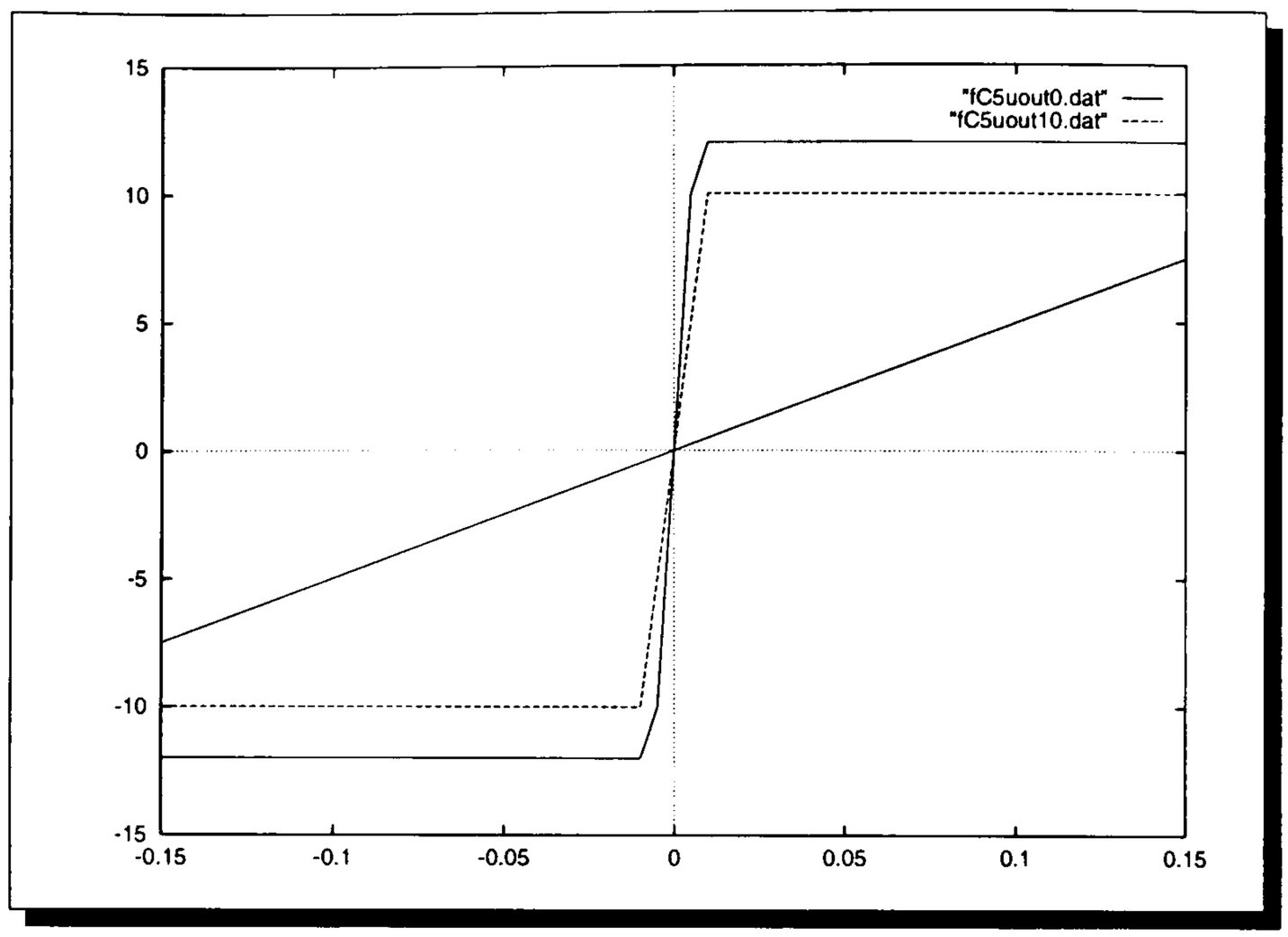


Figure 12.10: Result of Fuzzy Simulation of C5

To decide which of the two circuits meet the circuit paper prototype model best the real output from the database[1] and the fuzzy output at each α-cut-level is compared using the similarity measurement defined in Section 5.3.

The fuzzy curve, generated by the fuzzy simulation, consists of 61 temporal fuzzifying functions. Since the real output from the database is a non-fuzzy curve only the membership value corresponding to the real output has to be considered.

Fig. 12.11 shows the result of the fuzzy simulation (as displayed in Fig. 12.10) and the output function of the SPICE simulation (solid line). For circuit C5-A (Fig. 12.11) the similarity is:

$$S_{\text{temporal fuzzifying function}_1} = 0.1$$

$$S_{\text{temporal fuzzifying function}_2} = 0.1$$

$$S_{\text{temporal fuzzifying function}_3} = 0.1$$

$$\cdots$$

$$S_{\text{temporal fuzzifying function}_{29}} = 0.0$$

[1] determined by SPICE simulation

$$S_{\text{temporal fuzzifying function}_{30}} = 0.0$$

$$S_{\text{temporal fuzzifying function}_{31}} = 0.0$$

$$\cdots$$

$$S_{\text{temporal fuzzifying function}_{60}} = 0.3$$

$$S_{\text{temporal fuzzifying function}_{61}} = 0.3$$

$$S_{\text{fuzzy curve}} = \sum_{i=1}^{61} S_{\text{temporal fuzzifying function}_i} = 30.234$$

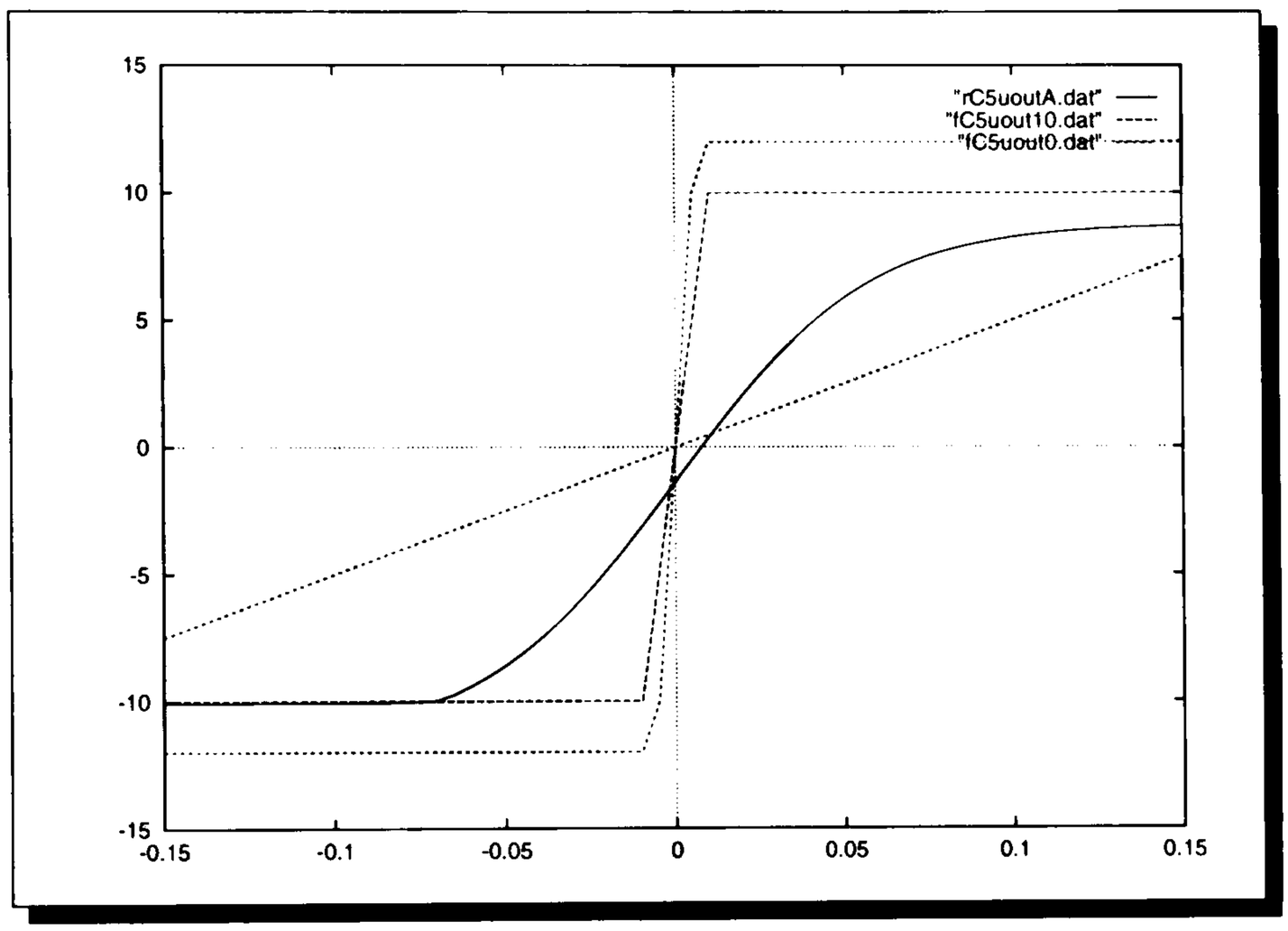


Figure 12.11: Result of Fuzzy Simulation of C5: A

Fig. 12.12 shows the result of the fuzzy simulation (as displayed in Fig. 12.10) and the output function of the SPICE simulation (solid line). For circuit C5-B (Fig. 12.12) the similarity is:

$$S_{\text{temporal fuzzifying function}_1} = 0.0$$

$$S_{\text{temporal fuzzifying function}_2} = 0.0$$

$$S_{\text{temporal fuzzifying function}_3} = 0.0$$

$$\cdots$$

$$S_{\text{temporal fuzzifying function}_{29}} = 1.0$$

$$S_{\text{temporal fuzzifying function}_{30}} = 1.0$$

$$S_{\text{temporal fuzzifying function}_{31}} = 1.0$$

$$\cdots$$

$$S_{\text{temporal fuzzifying function}_{60}} = 0.0$$

$$S_{\text{temporal fuzzifying function}_{61}} = 0.0$$

$$S_{\text{fuzzy curve}} = \sum_{i=1}^{61} S_{\text{temporal fuzzifying function}_i} = 33.535$$

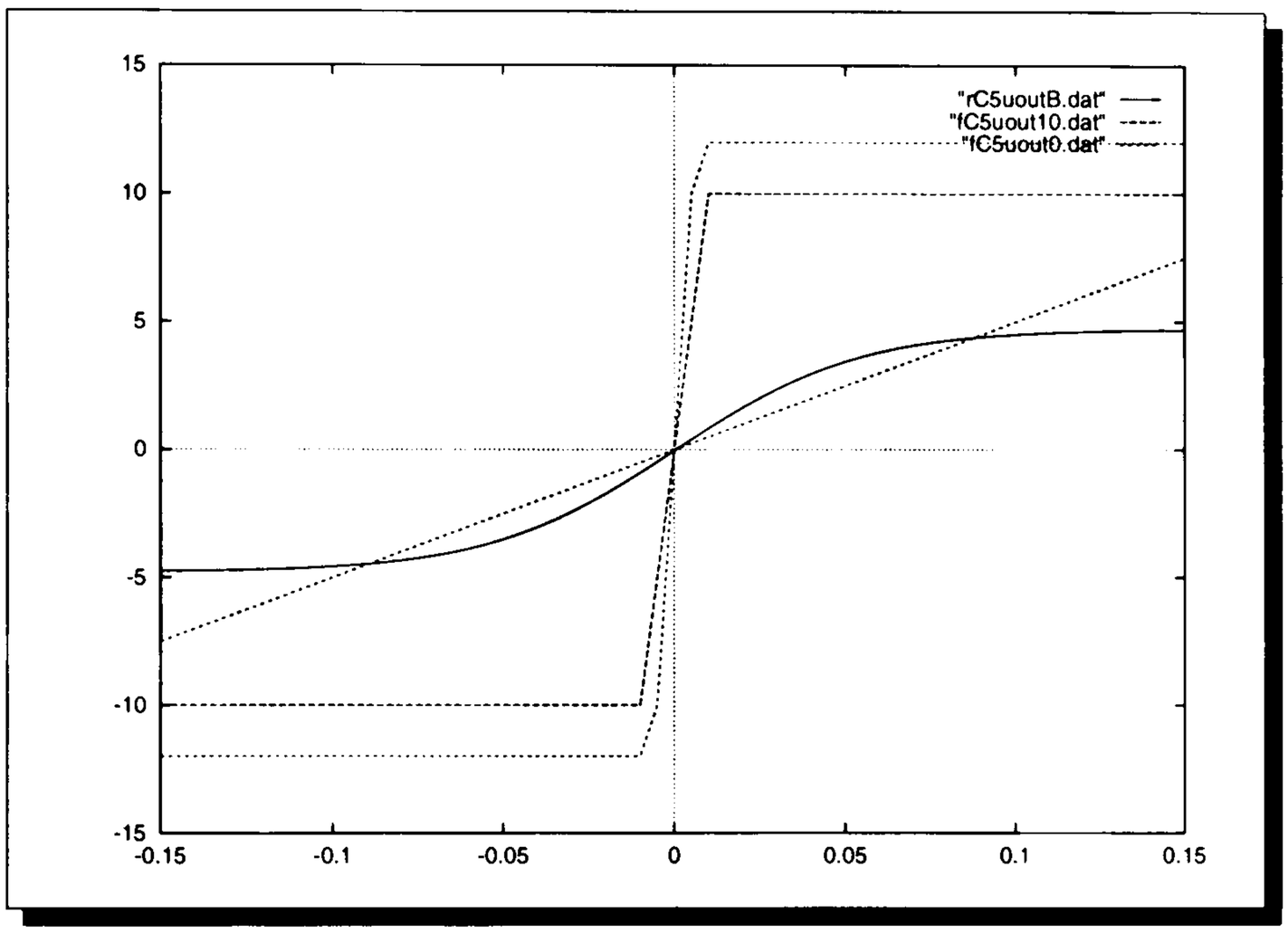


Figure 12.12: Result of Fuzzy Simulation of C5: B

The ordering of the two C5 circuits is C5B then C5A. Circuit C5B has the highest model correctness value.

12.5 Conclusion

Looking for a correct analogue circuit system in a database often fails, because they do not match exactly the specifications. The qualitative fuzzy approach evaluates the given systems of the database according to their specifications. The best fit

circuit is found. The search for the best suitable circuit is performed by a two step configuration-design approach. By doing a qualitative configuration-design, first all very unlikely circuits are thrown out of the possible circuit set for further investigation. Step two, fuzzy configuration-design finds out the best suitable circuit cell.

Chapter 13

Conclusions

This dissertation investigated modelling, simulation, and specification of imprecisely nonlinear systems. While much of the emphasis of this dissertation has been on the representation of imprecise signals and nonlinear models by the new fuzzy type Fuzzy Relation Memories, it has been shown that the combination of qualitative and fuzzy reasoning called qualitative fuzzy simulation can lead to a robust method for design, simulation, and data interpretation, especially in engineered analogue circuit design process.

Part I of this dissertation has presented a theoretical foundation for specifying, simulating, and modelling of nonlinear systems not known exactly. Specifying imprecise signals by **Fuzzy Relation Memories** allow us to build hierarchical structures by combining them by ordinary arithmetic operations. Modelling imprecise nonlinear systems by Fuzzy Relation Memories extends the known fuzzy systems. Besides the fuzzy rule-based modelling the equation-based modelling using differential equations is of importance for engineering domains. A new **Interactive Evolutionary Algorithm** approach, for solving ordinary differential equations whose initial values and equation parameters are not known exactly has been presented in this thesis. Pure qualitative simulation or fuzzy simulation, arithmetic operations with Fuzzy Relation Memories, and the solving of imprecise differential equations is summarized

by the concept of qualitative fuzzy simulation.

Part II is the application part using the findings and knowledge gathered in Part I. The most essential finding of the second part is a new framework for the design of analogue circuits called **High-Level Framework for Imprecise Analogue Circuit Design (IACD)**. The imprecision involved during the design, e.g. specification, modelling, searching in databases, is taken into account in this framework.

13.1 Contributions and Critical Review

13.1.1 Modelling with Fuzzy Relation Memories

- Specifying imprecise signals by Fuzzy Relation Memories using a graphical editor is natural for human designers. Analogue circuit designers sketch signal on paper visualizing the behaviour of a system. The FRMs are a generalization of such signals.

- Fuzzy Relation Memories are a logical extension of the known Fuzzy Associative Memories. Fuzzy Relation Memories can model nonlinear systems not known exactly.

- Arithmetic operations for Fuzzy Relation Memories have been developed which makes it possible to build up hierarchical models using Fuzzy Relation Memories.

- The piecewise linear build Fuzzy Relation Memories cause transition problems doing arithmetic operations, like differentiation.

13.1.2 Finding Solutions for Imprecise Differential Equations by the Interactive Evolutionary Algorithm Approach

- The Interactive Evolutionary Algorithm has shown its suitability for finding the minimum/maximum curve of the solution space for an imprecise differential equation.

- The approach is an indirect one, since not the chromosome set of the last generation is of importance but the optimal simulation results of the objective function during the evolutionary process.

- In spite of the correct simulation the numerical computation effort is very large.

13.1.3 High-Level Framework for Imprecise Analogue Circuit Design (IACD)

- There are many existing frameworks who support design engineers designing analogue circuits. Most of them have their own description languages or user interfaces to describe the system requirements. In the near future the VHDL-A standard [VHDL-A Standard 99, 1999] will become the most important hardware description language or continuous systems. For the simulation of analogue circuits SPICE [Nagel, 1973] is the most spread simulator. But neither the description language VHDL-A nor the simulator SPICE can handle at the moment the imprecision involved in specification and simulation. Therefore a new framework based on the qualitative fuzzy approach has been developed.

- In Part II of this thesis a framework has been developed that takes into account the imprecision involved in analogue circuit design. The three important

phases of the framework are:

  - **Phase I:** Circuit paper prototype design is the first step in the design of a circuit designer. Ideas and various experimental prototypes can be investigated.
  - **Phase II:** Qualitative configuration-design searches in a pre-simulated database for circuit cells that have the same qualitative behaviour as the circuit paper prototype model.
  - **Phase III:** Fuzzy configuration-design orders the found circuit cells of Phase II according to the ability to meet the circuit design specifications.

- The basis for all three phases is the qualitative fuzzy simulation for analyzing the designs. In general for qualitative fuzzy simulation more computational resources are needed. But the simulation results contain more information and gives the designer an idea what to change on the model to get the wanted behaviour.

- The major drawback is to have a pre-simulated database which stores the typical input and output data for the circuit. At present the developed framework has only be tested with the circuit database of Appendix C.

- Looking for the best circuit can be time consuming especially when the circuit has many inputs, since all permutations of the input must be tested with the model of the circuit paper prototype design stage.

13.2 Trends — Future Work

There have been identified three areas for possible extensions and future work of this research.

13.2.1 Common Interpretation of Membership Functions

In the future the imprecision involved in designing systems has to be taken into account. For the problem of membership function interpretation, which is the foundation of the qualitative fuzzy simulation, a standardization step for the membership function assignment problem should be introduced. To support team work design it is essential that the membership functions must be equally understood and interpreted by all the team members.

13.2.2 Experimentations with Real Circuit Cell Databases

The developed framework has to be proven to be successful with a large database of pre-simulated circuits. The used database are the circuits used for education of the fourth semester of an electronic engineering class at the university. A company's circuit cell database was not possible to use.

13.2.3 Extend Approach to Other Design Domains

Other engineering areas should be investigated for finding applications to use the introduced theory. Besides the electronic engineering design area, there are areas interesting to investigate, e.g. chemical engineering, economic systems, medicine, etc.

Appendix A

Fuzzy Reasoning Basics

A.1 Basic Operations

The classical mathematic carries out a number of operations. These operations can also be carried out using Triangular Fuzzy Numbers or Triangular Fuzzy Intervals. In the next view sections the basic operations

- addition,
- subtraction,
- multiplication, and
- division

on Triangular Fuzzy Numbers and Triangular Fuzzy Intervals are defined.

All operations have in common that they are done α-cut level by α-cut level. Let **A** and **B** be two Triangular Fuzzy Numbers (TFN) or two Triangular Fuzzy Intervals (TFI) and A_α and B_α be their intervals of confidence (support) for the level of presumption $\alpha, \alpha \in [0, 1]$.

Triangular Fuzzy Number

$$\tilde{C} = \tilde{A} \; operator \; \tilde{B} <=> (c, \gamma_c, \delta_c) = (a, \gamma_a, \delta_a) \; operator \; (b, \gamma_b, \delta_b) \qquad (A.1)$$

The operation is done level by level, so we can then write

Triangular Fuzzy Interval

$$\tilde{C} = \tilde{A}\ operator\ \tilde{B} <=> (c1, c2, \gamma_c, \delta_c) = (a1, a2, \gamma_a, \delta_a)\ operator\ (b1, b2, \gamma_b, \delta_b) \tag{A.2}$$

The operation is done level by level, so we can then write

$$\forall x, y, z \in R :$$

$$\mu_{A_\alpha operator B_\alpha}(z) = \bigvee_{z = x\ operator\ y} (\mu_{A_\alpha}(x) \wedge \mu_{B_\alpha}(y)) \tag{A.3}$$

The addition and the subtraction operator do not distinguish between non interactive operator, and strongly positive interactive operator. The different kind of interactions is well discussed in Chapter 5.

When using strongly positive interaction the two left/right boundary functions of **A** and **B** are used for calculating the new left/right boundary of **C**. Strongly negative interaction calculates the new left/right boundary of **C** using the left/right boundary function of **A** and one right/left boundary function of **B**. All possibilities of combining left and right boundary functions of **A** and **B** are calculated and the minimum/maximum result builds the new left/right boundary function of **C** when using non interactive operators.

A.2 Addition of Triangular Fuzzy Numbers

Definition A.75 (*Non Interactive Addition of Triangular Fuzzy Number*):

$$\begin{aligned} \tilde{C} &= \tilde{A}\tilde{+}\tilde{B} \\ &= (a, \gamma_a, \delta_a)\tilde{+}(b, \gamma_b, \delta_b) \\ &= (a + b, \gamma_a + \gamma_b, \delta_a + \delta_b) \end{aligned} \tag{A.4}$$

Definition A.76 (*Strongly Positive Interactive Addition of Triangular Fuzzy Number*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \overrightarrow{+} \tilde{B} \\ &= (a, \gamma_a, \delta_a) \overrightarrow{+} (b, \gamma_b, \delta_b) \\ &= (a+b, \gamma_a + \gamma_b, \delta_a + \delta_b)\end{aligned} \tag{A.5}$$

Definition A.77 (*Strongly Negative Interactive Addition of Triangular Fuzzy Number*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \overrightarrow{+} \tilde{B} \\ &= (a, \gamma_a, \delta_a) \overrightarrow{+} (b, \gamma_b, \delta_b) \\ &= (a+b, \gamma_a + \delta_b, \delta_a + \gamma_b)\end{aligned} \tag{A.6}$$

A.2.1 Example: Strongly Positive Addition of 2 Triangular Fuzzy Numbers

Another example is $\tilde{A} = (-1, 13, 3)$ and $\tilde{B} = (-2, 5, 16)$ added is $\tilde{A} \overrightarrow{+} \tilde{B} = (-3, 18, 19)$ shown in Fig. A.1.

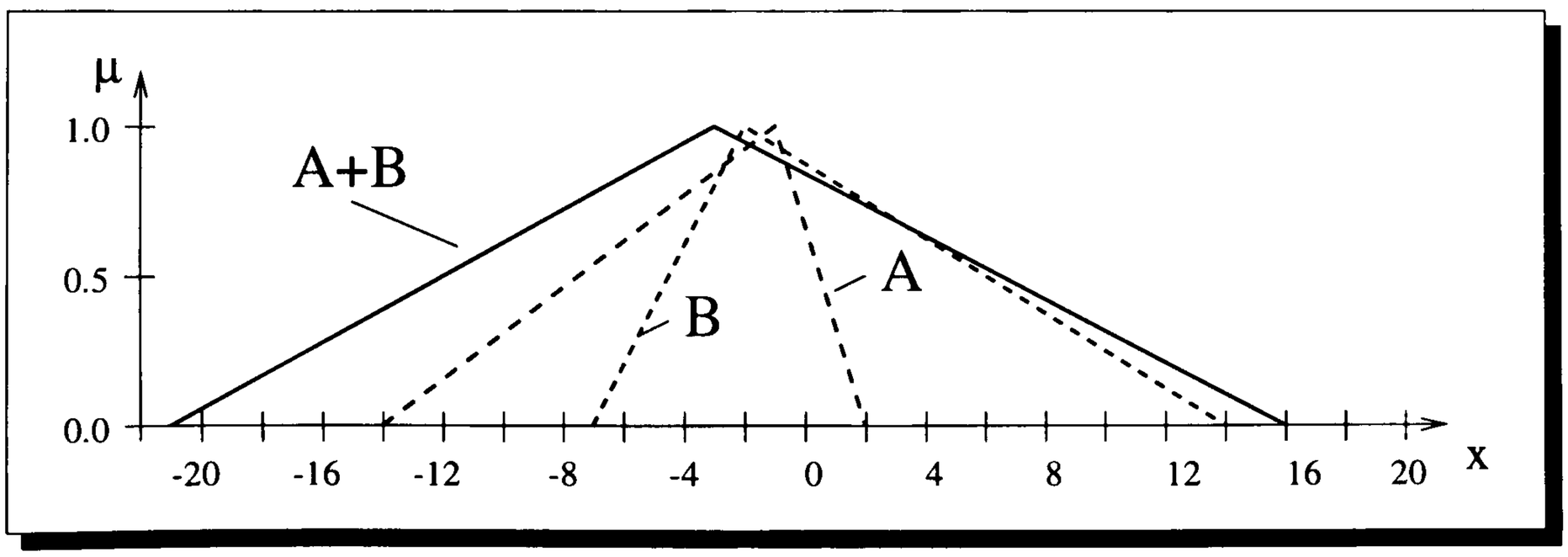


Figure A.1: Addition of Two Fuzzy Numbers

A.3 Addition of Triangular Fuzzy Intervals

Definition A.78 (*Non Interactive Addition of Triangular Fuzzy Intervals*):

$$\begin{aligned}\tilde{C} &= \tilde{A}\tilde{+}\tilde{B} \\ &= (a1, a2, \gamma_a, \delta_a)\tilde{+}(b1, b2, \gamma_b, \delta_b) \\ &= (a1 + b1, a2 + b2, \gamma_a + \gamma_b, \delta_a + \delta_b)\end{aligned} \tag{A.7}$$

Definition A.79 (*Strongly Positive Interactive Addition of Triangular Fuzzy Intervals*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \overrightarrow{+} \tilde{B} \\ &= (a1, a2, \gamma_a, \delta_a) \overrightarrow{+} (b1, b2, \gamma_b, \delta_b) \\ &= (a1 + b1, a2 + b2, \gamma_a + \gamma_b, \delta_a + \delta_b)\end{aligned} \tag{A.8}$$

Definition A.80 (*Strongly Negative Interactive Addition of Triangular Fuzzy Intervals*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \overrightarrow{\leftarrow} \tilde{B} \\ &= (a1, a2, \gamma_a, \delta_a) \overrightarrow{+} (b1, b2, \gamma_b, \delta_b) \\ &= (a1 + b2, a2 + b1, \gamma_a + \delta_b, \delta_a + \gamma_b)\end{aligned} \tag{A.9}$$

A.3.1 Example: Strongly Positive Interactive Addition of 2 Triangular Fuzzy Intervals

Given the two Triangular Fuzzy Intervals $\tilde{A} = (-2, -1, 3, 3)$ and $\tilde{B} = (-1, 1, 1, 6)$. Strongly positive interactive addition results into $\tilde{A} \overrightarrow{+} \tilde{B} = (-3, 0, 4, 9)$ shown in Fig. A.2.

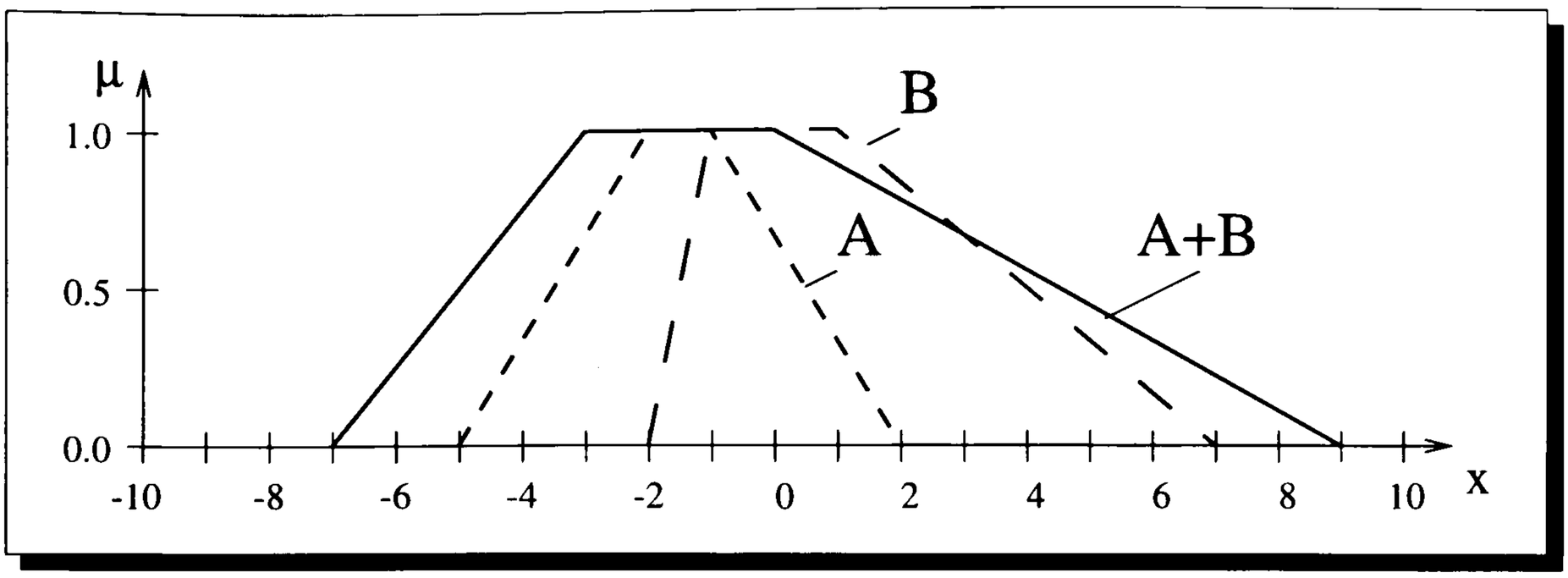


Figure A.2: Addition of Two Fuzzy Intervals

A.4 Subtraction of Triangular Fuzzy Numbers

Definition A.81 (*Non Interactive Subtraction of Triangular Fuzzy Number*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \tilde{-} \tilde{B} \\ &= (a, \gamma_a, \delta_a) \tilde{-} (b, \gamma_b, \delta_b) \\ &= (a-b, \gamma_a + \delta_b, \delta_a + \gamma_b)\end{aligned} \tag{A.10}$$

Definition A.82 (*Strongly Positive Interactive Subtraction of Triangular Fuzzy Number*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \overset{\rightrightarrows}{-} \tilde{B} \\ &= (a, \gamma_a, \delta_a) \overset{\rightrightarrows}{-} (b, \gamma_b, \delta_b) \\ &= (a-b, \gamma_a + \delta_b, \delta_a + \gamma_b)\end{aligned} \tag{A.11}$$

Definition A.83 (*Strongly Negative Interactive Subtraction of Triangular Fuzzy Number*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \overset{\rightleftarrows}{-} \tilde{B} \\ &= (a, \gamma_a, \delta_a) \overset{\rightleftarrows}{-} (b, \gamma_b, \delta_b) \\ &= (a-b, \gamma_a + \gamma_b, \delta_a + \delta_b)\end{aligned} \tag{A.12}$$

A.4.1 Example: Strongly Positive Subtraction of 2 Triangular Fuzzy Numbers

$\tilde{A} = (-4, 18, 24)$ and $\tilde{B} = (-2, 14, 4)$ subtracted is $\tilde{A} \overset{\Rightarrow}{-} \tilde{B} = (-2, 22, 38)$ shown in Fig. A.3.

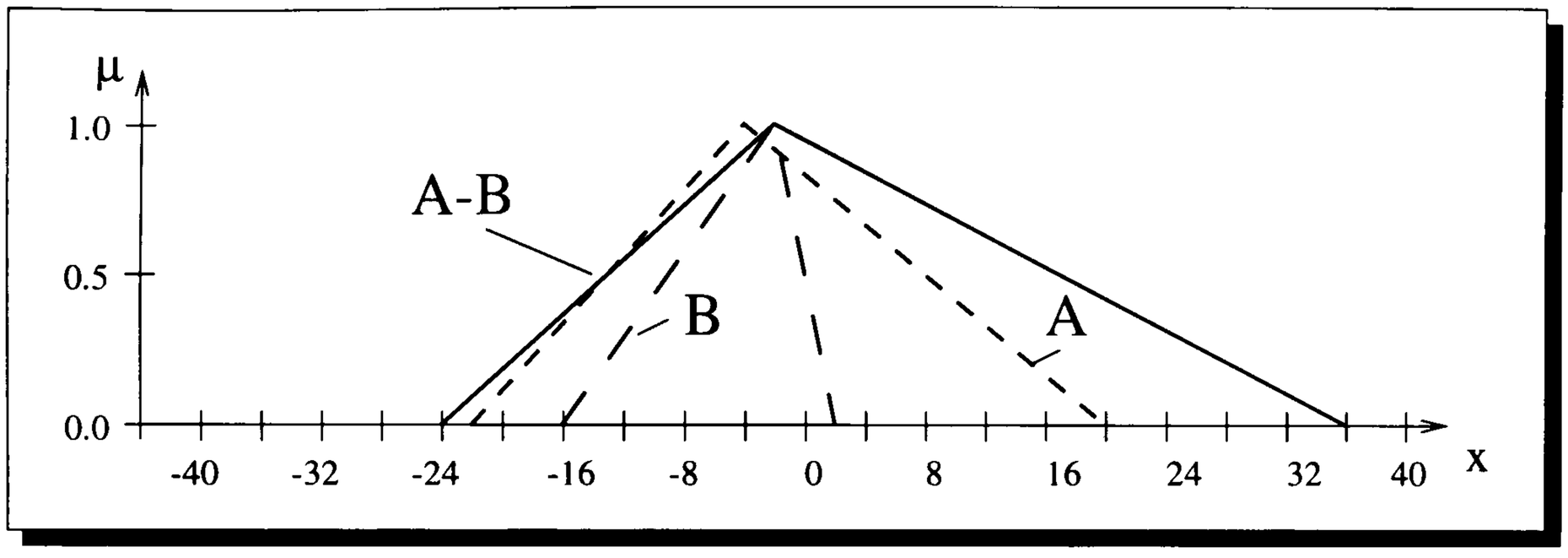


Figure A.3: Subtraction of Two Fuzzy Numbers

A.5 Subtraction of Triangular Fuzzy Intervals

Definition A.84 (*Non Interactive Subtraction of Triangular Fuzzy Intervals*):

$$\begin{aligned} \tilde{C} &= \tilde{A} \tilde{-} \tilde{B} \\ &= (a1, a2, \gamma_a, \delta_a) \tilde{-} (b1, b2, \gamma_b, \delta_b) \\ &= (a1 - b2, a2 - b1, \gamma_a + \delta_b, \delta_a + \gamma_b) \end{aligned} \tag{A.13}$$

Definition A.85 (*Strongly Positive Interactive Subtraction of Triangular Fuzzy Intervals*):

$$\begin{aligned} C &= A - B \\ &= (a1, a2, \gamma_a, \delta_a) \overset{\Rightarrow}{-} (b1, b2, \gamma_b, \delta_b) \\ &= (a1 - b2, a2 - b1, \gamma_a + \delta_b, \delta_a + \gamma_b) \end{aligned} \tag{A.14}$$

Definition A.86 (*Strongly Negative Interactive Subtraction of Triangular Fuzzy Intervals*):

$$\begin{aligned}\tilde{C} &= \tilde{A} \overset{\rightleftarrows}{-} \tilde{B} \\ &= (a1, a2, \gamma_a, \delta_a) \overset{\rightleftarrows}{-} (b1, b2, \gamma_b, \delta_b) \\ &= (a1 - b1, a2 - b2, \gamma_a + \gamma_b, \delta_a + \delta_b)\end{aligned} \tag{A.15}$$

A.5.1 Example: Strongly Positive Interactive Subtraction of 2 Triangular Fuzzy Intervals

Given the two Triangular Fuzzy Intervals $\tilde{A} = (-3, 2, 4, 7)$ and $\tilde{B} = (-2, -1, 3, 3)$. Strongly positive interactive addition results into $\tilde{A} \overset{\rightrightarrows}{-} \tilde{B} = (-2, 4, 7, 10)$ shown in Fig. A.4.

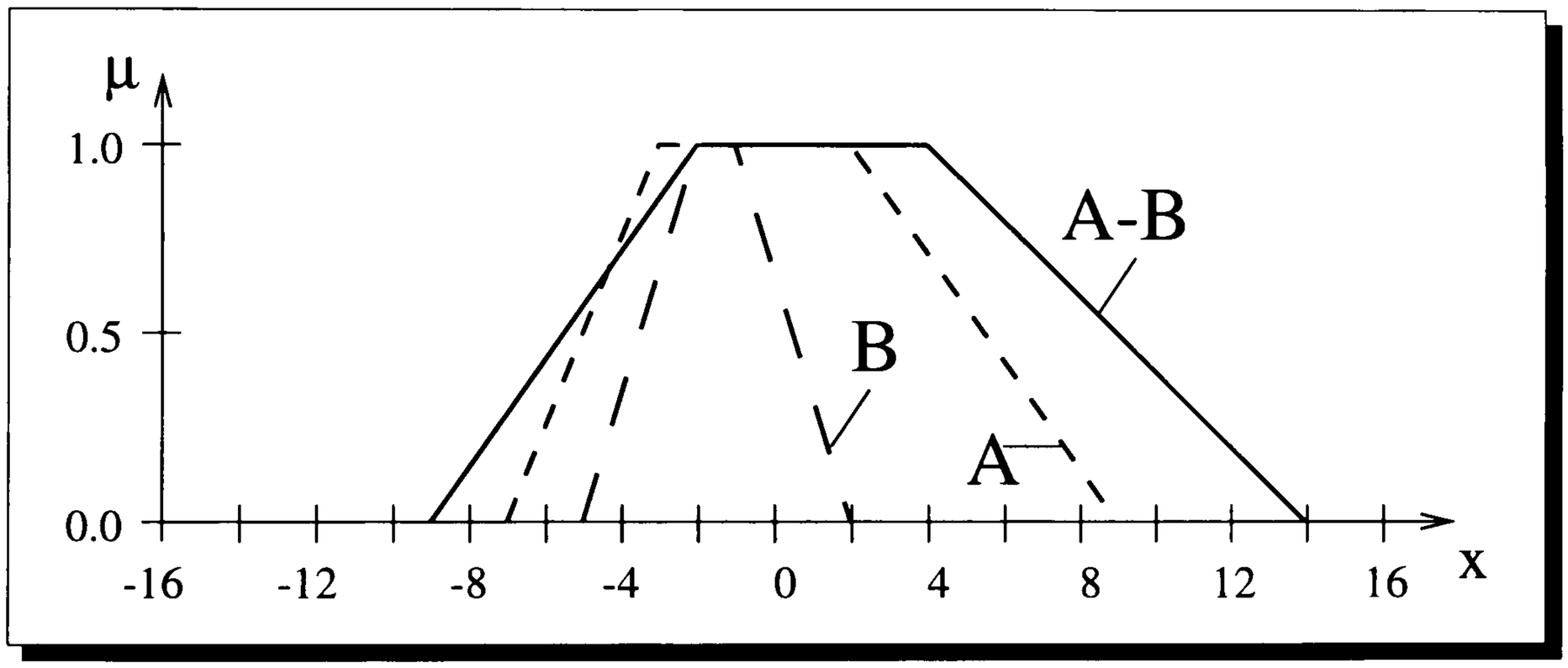


Figure A.4: Subtraction of Two Fuzzy Numbers

A.6 Multiplication of Triangular Fuzzy Numbers

If two fuzzy values are multiplied, the left and right boundary of the multiplication are of quadratic nature. For numerical convenience the boundary of the evaluated fuzzy values are linearized by the following:

Definition A.87 (*Non Interaction Multiplication of Triangular Fuzzy Numbers*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \tilde{*} \tilde{B}_{TFN} <=> (c, \gamma_c, \delta_c) = (a, \gamma_a, \delta_a) \tilde{*} (b, \gamma_b, \delta_b)$$

approximated :

$$\begin{aligned} c &= a * b \\ \gamma_c &= c - min[((a-\gamma_a)*(b-\gamma_b)), ((a-\gamma_a)*(b+\delta_b)), \\ &\quad ((a+\delta_a)*(b-\gamma_b)), ((a+\delta_a)*(b+\delta_b))] \\ \delta_c &= -c + max[((a-\gamma_a)*(b-\gamma_b)), ((a-\gamma_a)*(b+\delta_b)), \\ &\quad ((a+\delta_a)*(b-\gamma_b)), ((a+\delta_a)*(b+\delta_b))] \end{aligned} \tag{A.16}$$

Definition A.88 (*Strongly Positive Interactive Multiplication of Triangular Fuzzy Numbers*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \overset{\rightrightarrows}{*} \tilde{B}_{TFN} <=> (c, \gamma_c, \delta_c) = (a, \gamma_a, \delta_a) \overset{\rightrightarrows}{*} (b, \gamma_b, \delta_b)$$

approximated :

$$\begin{aligned} c &= a * b \\ \gamma_c &= c - min[((a-\gamma_a)*(b-\gamma_b)), ((a+\delta_a)*(b+\delta_b))] \\ \delta_c &= -c + max[((a-\gamma_a)*(b-\gamma_b)), ((a+\delta_a)*(b+\delta_b))] \end{aligned} \tag{A.17}$$

Definition A.89 (*Strongly Negative Interactive Multiplication of Triangular Fuzzy Numbers*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \overset{\rightleftarrows}{*} \tilde{B}_{TFN} <=> (c, \gamma_c, \delta_c) = (a, \gamma_a, \delta_a) \overset{\rightleftarrows}{*} (b, \gamma_b, \delta_b)$$

approximated :

$$\begin{aligned} c &= a * b \\ \gamma_c &= c - min[((a-\gamma_a)*(b+\delta_b)), ((a+\delta_a)*(b-\gamma_b))] \\ \delta_c &= -c + max[((a-\gamma_a)*(b+\delta_b)), ((a+\delta_a)*(b-\gamma_b))] \end{aligned} \tag{A.18}$$

A.6.1 Example: Non Interactive Multiplication of Triangular Fuzzy Numbers

$A = [-6, 1, 8]$ and $B = [2, 1, 7]$ multiplied is $A\tilde{*}B = [-12, 51, 30]$ shown in Fig. A.5.

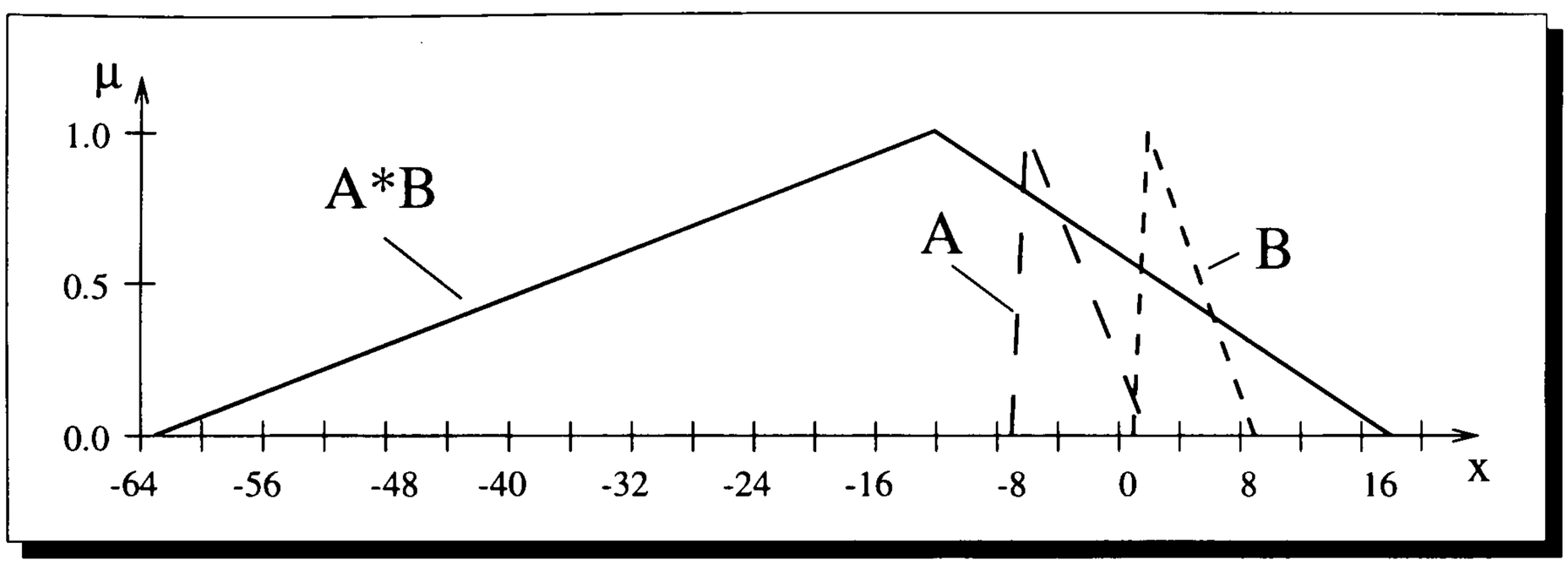


Figure A.5: Multiplication of two Triangular Fuzzy Numbers

A.7 Multiplication of Triangular Fuzzy Intervals

Definition A.90 (*Non Interaction Multiplication of Triangular Fuzzy Interval*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \tilde{*} \tilde{B}_{TFN} <=> (c1, c2, \gamma_c, \delta_c) = (a1, a2, \gamma_a, \delta_a) \tilde{*} (b1, b2, \gamma_b, \delta_b)$$

$approximated:$

$$\begin{aligned}
c1 &= min[(a1 * b1), \\
&\quad (a1 * b2), \\
&\quad (a2 * b1), \\
&\quad (a2 * b2)] \\
c2 &= max[(a1 * b1), \\
&\quad (a1 * b2), \\
&\quad (a2 * b1), \\
&\quad (a2 * b2)] \\
\gamma_c &= c1 - min[((a1 - \gamma_a) * (b1 - \gamma_b)), ((a1 - \gamma_a) * (b2 + \delta_b)), \\
&\quad ((a2 + \delta_a) * (b1 - \gamma_b)), ((a2 + \delta_a) * (b2 + \delta_b))] \\
\delta_c &= -c2 + max[((a1 - \gamma_a) * (b1 - \gamma_b)), ((a1 - \gamma_a) * (b2 + \delta_b)), \\
&\quad ((a2 + \delta_a) * (b1 - \gamma_b)), ((a2 + \delta_a) * (b2 + \delta_b))]
\end{aligned} \tag{A.19}$$

Definition A.91 (*Strongly Positive Interactive Multiplication of Triangular Fuzzy Interval*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \overrightarrow{*} \tilde{B}_{TFN} <=> (c1, c2, \gamma_c, \delta_c) = (a1, a2, \gamma_a, \delta_a) \overrightarrow{*} (b1, b2, \gamma_b, \delta_b)$$

approximated :

$$c1 = min[a1 * b1, a2 * b2]$$

$$c2 = max[a1 * b1, a2 * b2]$$

$$\gamma_c = c1 - min[((a1 - \gamma_a) * (b1 - \gamma_b)), ((a2 + \delta_a) * (b2 + \delta_b))]$$

$$\delta_c = -c2 + max[((a1 - \gamma_a) * (b1 - \gamma_b)), ((a2 + \delta_a) * (b2 + \delta_b))] \quad (A.20)$$

Definition A.92 (*Strongly Negative Interactive Multiplication of Triangular Fuzzy Interval*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \overrightarrow{\leftarrow *} \tilde{B}_{TFN} <=> (c1, c2, \gamma_c, \delta_c) = (a1, a2, \gamma_a, \delta_a) \overrightarrow{\leftarrow *} (b1, b2, \gamma_b, \delta_b)$$

approximated :

$$c1 = min[a1 * b2, a2 * b1]$$

$$c2 = max[a1 * b2, a2 * b1]$$

$$\gamma_c = c1 - min[((a1 - \gamma_a) * (b2 + \delta_b)), ((a2 + \delta_a) * (b1 - \gamma_b))]$$

$$\delta_c = -c2 + max[((a1 - \gamma_a) * (b2 + \delta_b)), ((a2 + \delta_a) * (b1 - \gamma_b))] \quad (A.21)$$

A.7.1 Example: Non Interactive Multiplication of Triangular Fuzzy Intervals

Assume the two fuzzy numbers $\tilde{A} = [-2, -1, 1, 3]$ and $\tilde{B} = [2, 3, 1, 1]$. The multiplication of the two fuzzy number results into $\tilde{A} \tilde{*} \tilde{B} = [-6, -2, 6, 10]$ shown in Fig. A.6

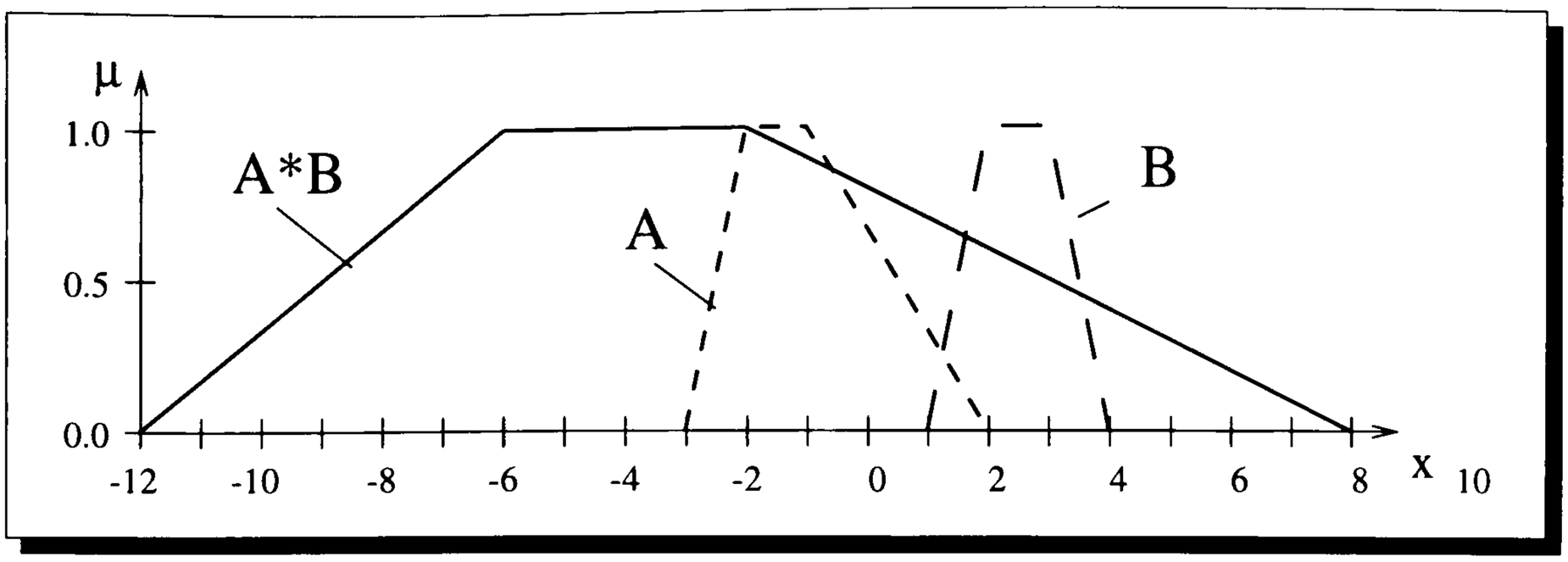


Figure A.6: Multiplication of two Triangular Fuzzy Intervals

A.8 Division of Triangular Fuzzy Numbers

If two fuzzy values are divided, the left and right boundary of the division are nonlinear. For numerical convenience the boundary of the evaluated fuzzy values are linearized by the following:

Definition A.93 (*Non Interactive Division of Triangular Fuzzy Numbers*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \tilde{/} \tilde{B}_{TFN} <=> (c, \gamma_c, \delta_c) = (a, \gamma_a, \delta_a) \tilde{/} (b, \gamma_b, \delta_b)$$

approximated :

$$\begin{aligned} c &= \frac{a}{b} \\ \gamma_c &= c - min[\frac{a-\gamma_a}{b-\gamma_b}, \frac{a-\gamma_a}{b+\delta_b}, \frac{a+\delta_a}{b-\gamma_b}, \frac{a+\delta_a}{b+\delta_b}] \\ \delta_c &= -c + max[\frac{a-\gamma_a}{b-\gamma_b}, \frac{a-\gamma_a}{b+\delta_b}, \frac{a+\delta_a}{b-\gamma_b}, \frac{a+\delta_a}{b+\delta_b}] \end{aligned} \tag{A.22}$$

Definition A.94 (*Strongly Positive Interactive Division of Triangular Fuzzy Numbers*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \overrightarrow{/} \tilde{B}_{TFN} <=> (c, \gamma_c, \delta_c) = (a, \gamma_a, \delta_a) \overrightarrow{/} (b, \gamma_b, \delta_b)$$

approximated :

$$c = \frac{a}{b}$$
$$\gamma_c = c - min[\frac{a - \gamma_a}{b - \gamma_b}, \frac{a + \delta_a}{b + \delta_b}]$$
$$\delta_c = -c + max[\frac{a - \gamma_a}{b - \gamma_b}, \frac{a + \delta_a}{b + \delta_b}] \tag{A.23}$$

Definition A.95 (*Strongly Negative Interactive Division of Triangular Fuzzy Numbers*):

$$\tilde{C}_{TFN} = \tilde{A}_{TFN} \overleftarrow{/} \tilde{B}_{TFN} <=> (c, \gamma_c, \delta_c) = (a, \gamma_a, \delta_a) \overleftarrow{/} (b, \gamma_b, \delta_b)$$

approximated :

$$c = \frac{a}{b}$$
$$\gamma_c = c - min[\frac{a - \gamma_a}{b + \delta_b}, \frac{a + \delta_a}{b - \gamma_b}]$$
$$\delta_c = -c + max[\frac{a - \gamma_a}{b + \delta_b}, \frac{a + \delta_a}{b - \gamma_b}] \tag{A.24}$$

A.8.1 Example: Non Interactive Division of Triangular Fuzzy Intervals

$\tilde{A} = [-8, 4, 16]$ and $\tilde{B} = [2, 1, 8]$ divided is $\frac{\tilde{A}}{\tilde{B}} = [-4, 8, 12]$ shown in Fig. A.7.

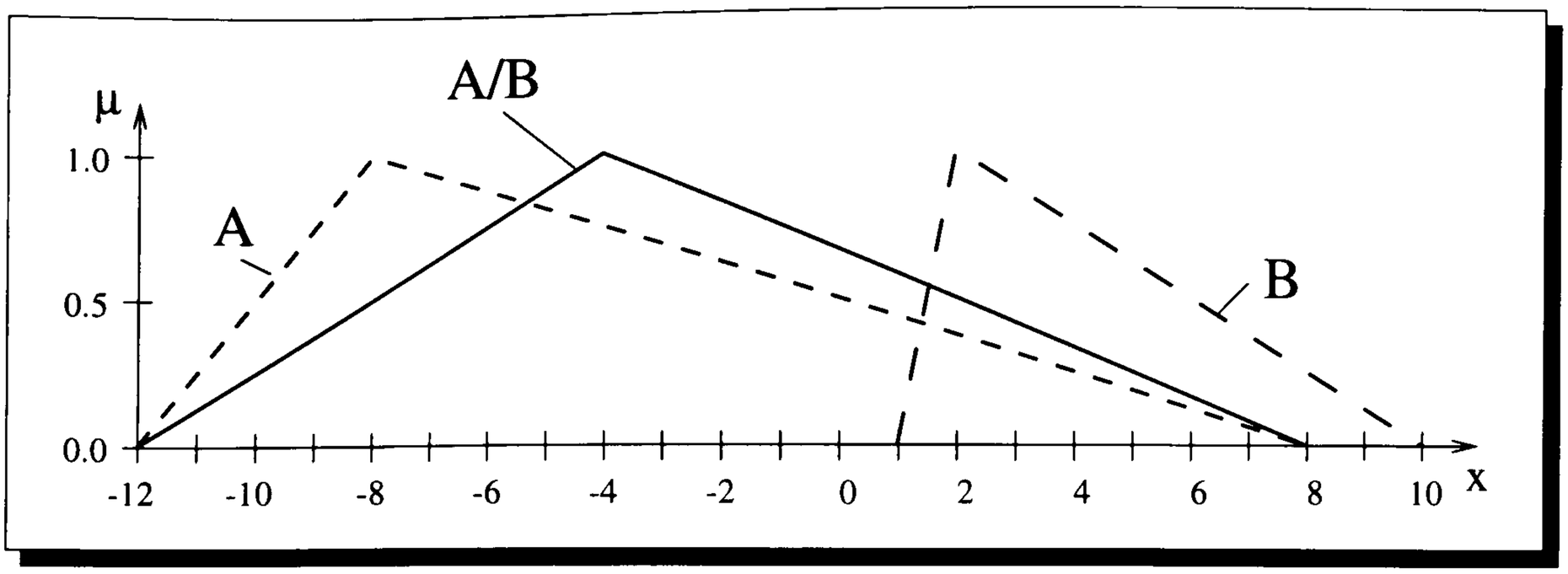


Figure A.7: Division of Two Fuzzy Numbers

A.9 Division of Triangular Fuzzy Intervals

Definition A.96 (*Non Interactive Division of Triangular Fuzzy Intervals*): Two triangular fuzzy intervals are divided and approximated by the formula:

$$\tilde{C}_{TFI} = \tilde{A}_{TFI} \tilde{/} \tilde{B}_{TFI} <=> (c1, c2, \gamma_c, \delta_c) = (a1, a2, \gamma_a, \delta_a) \tilde{/} (b1, b2, \gamma_b, \delta_b)$$

approximated :

$$\begin{aligned}
c1 &= min[\frac{a1}{b1}, \frac{a1}{b2}, \frac{a2}{b1}, \frac{a2}{b2}] \\
c2 &= max[\frac{a1}{b1}, \frac{a1}{b2}, \frac{a2}{b1}, \frac{a2}{b2}] \\
\gamma_c &= c1 - min[\frac{a1 - \gamma_a}{b1 - \gamma_b}, \frac{a1 - \gamma_a}{b2 + \delta_b}, \frac{a2 + \delta_a}{b1 - \gamma_b}, \frac{a2 + \delta_a}{b2 + \delta_b}] \\
\delta_c &= -c2 + max[\frac{a1 - \gamma_a}{b1 - \gamma_b}, \frac{a1 - \gamma_a}{b2 + \delta_b}, \frac{a2 + \delta_a}{b1 - \gamma_b}, \frac{a2 + \delta_a}{b1 + \delta_b}]
\end{aligned} \tag{A.25}$$

Definition A.97 (*Strongly Positive Interactive Division of Triangular Fuzzy Interval*):

$$\tilde{C}_{TFI} = \tilde{A}_{TFI} \overrightarrow{/} \tilde{B}_{TFI} <=> (c1, c2, \gamma_c, \delta_c) = (a1, a2, \gamma_a, \delta_a) \overrightarrow{/} (b1, b2, \gamma_b, \delta_b)$$

approximated :

$$\begin{aligned} c1 &= min[\frac{a1}{b1}, \frac{a2}{b2}] \\ c2 &= max[\frac{a1}{b1}, \frac{a2}{b2}] \\ \gamma_c &= c1 - min[\frac{a1 - \gamma_a}{b1 - \gamma_b}, \frac{a2 + \delta_a}{b2 + \delta_b}] \\ \delta_c &= -c2 + max[\frac{a1 - \gamma_a}{b1 - \gamma_b}, \frac{a2 + \delta_a}{b1 + \delta_b}] \end{aligned} \tag{A.26}$$

Definition A.98 (*Strongly Negative Interactive Division of Triangular Fuzzy Interval*):

$$\tilde{C}_{TFI} = \tilde{A}_{TFI} \overleftarrow{/} \tilde{B}_{TFI} <=> (c1, c2, \gamma_c, \delta_c) = (a1, a2, \gamma_a, \delta_a) \overleftarrow{/} (b1, b2, \gamma_b, \delta_b)$$

approximated :

$$\begin{aligned} c1 &= min[\frac{a1}{b2}, \frac{a2}{b1}] \\ c2 &= max[\frac{a1}{b2}, \frac{a2}{b1}] \\ \gamma_c &= c1 - min[\frac{a1 - \gamma_a}{b2 + \delta_b}, \frac{a2 + \delta_a}{b1 - \gamma_b}] \\ \delta_c &= -c2 + max[\frac{a1 - \gamma_a}{b2 + \delta_b}, \frac{a2 + \delta_a}{b1 - \gamma_b}] \end{aligned} \tag{A.27}$$

A.9.1 Example: Non Interactive Division of Triangular Fuzzy Intervals

Assume the two fuzzy numbers $\tilde{A} = [-6, -2, 6, 10]$ and $\tilde{B} = [2, 3, 1, 1]$. The division of the two fuzzy intervals results into $\frac{\tilde{A}}{\tilde{B}} = [-3, -1, 9, 9]$ shown in Fig. A.8

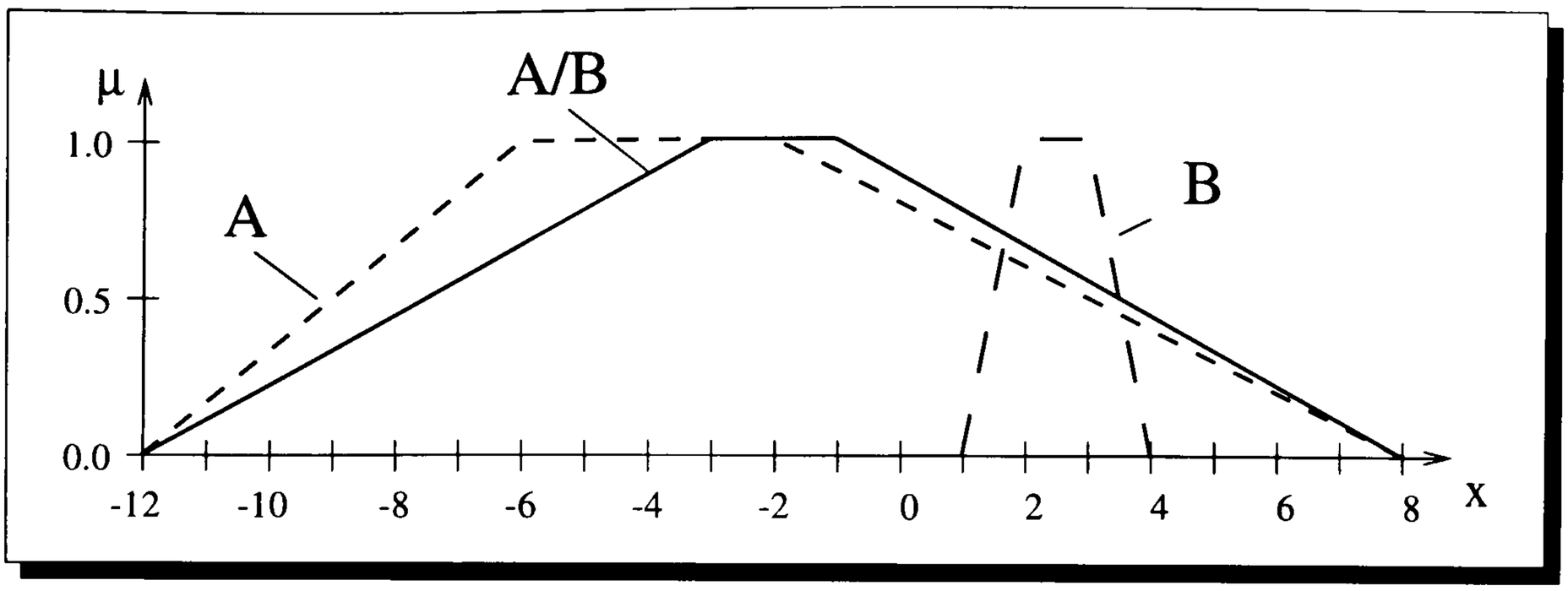


Figure A.8: Division of Two Fuzzy Numbers

A.10 Table of Triangular Fuzzy Intervals

$(6, 8, 2, 2)$	NonPlus	$(2, 3, 1, 1)$	$=$	$(8, 11, 3, 3)$
$(6, 8, 2, 2)$	StrongPosPlus	$(2, 3, 1, 1)$	$=$	$(8, 11, 3, 3)$
$(6, 8, 2, 2)$	StrongNegPlus	$(2, 3, 1, 1)$	$=$	$(9, 10, 3, 3)$
$(6, 8, 2, 2)$	NonMinus	$(2, 3, 1, 1)$	$=$	$(3, 6, 3, 3)$
$(6, 8, 2, 2)$	StrongPosMinus	$(2, 3, 1, 1)$	$=$	$(3, 6, 3, 3)$
$(6, 8, 2, 2)$	StrongNegMinus	$(2, 3, 1, 1)$	$=$	$(4, 5, 3, 3)$
$(6, 8, 2, 2)$	NonMul	$(2, 3, 1, 1)$	$=$	$(12, 24, 8, 16)$
$(6, 8, 2, 2)$	StrongPosMul	$(2, 3, 1, 1)$	$=$	$(12, 24, 8, 16)$
$(6, 8, 2, 2)$	StrongNegMul	$(2, 3, 1, 1)$	$=$	$(16, 18, 6, -2)$
$(6, 8, 2, 2)$	NonDiv	$(2, 3, 1, 1)$	$=$	$(2, 4, 1, 6)$
$(6, 8, 2, 2)$	StrongPosDiv	$(2, 3, 1, 1)$	$=$	$(2.67, 3, 0.167, 1)$
$(6, 8, 2, 2)$	StrongNegDiv	$(2, 3, 1, 1)$	$=$	$(2, 4, 1, 6)$

$(-8,-6,2,2)$	NonPlus	$(-3,-2,1,1)$	$=$	$(-11,-8,3,3)$
$(-8,-6,2,2)$	StrongPosPlus	$(-3,-2,1,1)$	$=$	$(-11,-8,3,3)$
$(-8,-6,2,2)$	StrongNegPlus	$(-3,-2,1,1)$	$=$	$(-10,-9,3,3)$
$(-8,-6,2,2)$	NonMinus	$(-3,-2,1,1)$	$=$	$(-6,-3,3,3)$
$(-8,-6,2,2)$	StrongPosMinus	$(-3,-2,1,1)$	$=$	$(-6,-3,3,3)$
$(-8,-6,2,2)$	StrongNegMinus	$(-3,-2,1,1)$	$=$	$(-5,-4,3,3)$
$(-8,-6,2,2)$	NonMul	$(-3,-2,1,1)$	$=$	$(12,24,8,16)$
$(-8,-6,2,2)$	StrongPosMul	$(-3,-2,1,1)$	$=$	$(12,24,8,16)$
$(-8,-6,2,2)$	StrongNegMul	$(-3,-2,1,1)$	$=$	$(16,18,6,-2)$
$(-8,-6,2,2)$	NonDiv	$(-3,-2,1,1)$	$=$	$(2,4,1,6)$
$(-8,-6,2,2)$	StrongPosDiv	$(-3,-2,1,1)$	$=$	$(2.66667,3,0.166667,1)$
$(-8,-6,2,2)$	StrongNegDiv	$(-3,-2,1,1)$	$=$	$(2,4,1,6)$

$(6,8,2,2)$	NonPlus	$(-3,-2,1,1)$	$=$	$(3,6,3,3)$
$(6,8,2,2)$	StrongPosPlus	$(-3,-2,1,1)$	$=$	$(3,6,3,3)$
$(6,8,2,2)$	StrongNegPlus	$(-3,-2,1,1)$	$=$	$(4,5,3,3)$
$(6,8,2,2)$	NonMinus	$(-3,-2,1,1)$	$=$	$(8,11,3,3)$
$(6,8,2,2)$	StrongPosMinus	$(-3,-2,1,1)$	$=$	$(8,11,3,3)$
$(6,8,2,2)$	StrongNegMinus	$(-3,-2,1,1)$	$=$	$(9,10,3,3)$
$(6,8,2,2)$	NonMul	$(-3,-2,1,1)$	$=$	$(-24,-12,16,8)$
$(6,8,2,2)$	StrongPosMul	$(-3,-2,1,1)$	$=$	$(-18,-16,-2,6)$
$(6,8,2,2)$	StrongNegMul	$(-3,-2,1,1)$	$=$	$(-24,-12,16,8)$
$(6,8,2,2)$	NonDiv	$(-3,-2,1,1)$	$=$	$(-4,-2,6,1)$
$(6,8,2,2)$	StrongPosDiv	$(-3,-2,1,1)$	$=$	$(-4,-2,6,1)$
$(6,8,2,2)$	StrongNegDiv	$(-3,-2,1,1)$	$=$	$(-3,-2.67,1,0.167)$

$(-8,-6,2,2)$	NonPlus	$(2,3,1,1)$	$=$	$(-6,-3,3,3)$
$(-8,-6,2,2)$	StrongPosPlus	$(2,3,1,1)$	$=$	$(-6,-3,3,3)$
$(-8,-6,2,2)$	StrongNegPlus	$(2,3,1,1)$	$=$	$(-5,-4,3,3)$
$(-8,-6,2,2)$	NonMinus	$(2,3,1,1)$	$=$	$(-11,-8,3,3)$
$(-8,-6,2,2)$	StrongPosMinus	$(2,3,1,1)$	$=$	$(-11,-8,3,3)$
$(-8,-6,2,2)$	StrongNegMinus	$(2,3,1,1)$	$=$	$(-10,-9,3,3)$
$(-8,-6,2,2)$	NonMul	$(2,3,1,1)$	$=$	$(-24,-12,16,8)$
$(-8,-6,2,2)$	StrongPosMul	$(2,3,1,1)$	$=$	$(-18,-16,-2,6)$
$(-8,-6,2,2)$	StrongNegMul	$(2,3,1,1)$	$=$	$(-24,-12,16,8)$
$(-8,-6,2,2)$	NonDiv	$(2,3,1,1)$	$=$	$(-4,-2,6,1)$
$(-8,-6,2,2)$	StrongPosDiv	$(2,3,1,1)$	$=$	$(-4,-2,6,1)$
$(-8,-6,2,2)$	StrongNegDiv	$(2,3,1,1)$	$=$	$(-3,-2.67,1,0.17)$

$(-1,1,2,2)$	NonPlus	$(2,3,1,1)$	$=$	$(1,4,3,3)$
$(-1,1,2,2)$	StrongPosPlus	$(2,3,1,1)$	$=$	$(1,4,3,3)$
$(-1,1,2,2)$	StrongNegPlus	$(2,3,1,1)$	$=$	$(2,3,3,3)$
$(-1,1,2,2)$	NonMinus	$(2,3,1,1)$	$=$	$(-4,-1,3,3)$
$(-1,1,2,2)$	StrongPosMinus	$(2,3,1,1)$	$=$	$(-4,-1,3,3)$
$(-1,1,2,2)$	StrongNegMinus	$(2,3,1,1)$	$=$	$(-3,-2,3,3)$
$(-1,1,2,2)$	NonMul	$(2,3,1,1)$	$=$	$(-3,3,9,9)$
$(-1,1,2,2)$	StrongPosMul	$(2,3,1,1)$	$=$	$(-2,3,1,9)$
$(-1,1,2,2)$	StrongNegMul	$(2,3,1,1)$	$=$	$(-3,2,9,1)$
$(-1,1,2,2)$	NonDiv	$(2,3,1,1)$	$=$	$(-0.5,0.5,2.5,2.5)$
$(-1,1,2,2)$	StrongPosDiv	$(2,3,1,1)$	$=$	$(-0.5,0.33,2.5,0.417)$
$(-1,1,2,2)$	StrongNegDiv	$(2,3,1,1)$	$=$	$(-0.33,0.5,0.417,2.5)$

$(-1,1,2,2)$	NonPlus	$(-3,-2,1,1)$	$=$	$(-4,-1,3,3)$
$(-1,1,2,2)$	StrongPosPlus	$(-3,-2,1,1)$	$=$	$(-4,-1,3,3)$
$(-1,1,2,2)$	StrongNegPlus	$(-3,-2,1,1)$	$=$	$(-3,-2,3,3)$
$(-1,1,2,2)$	NonMinus	$(-3,-2,1,1)$	$=$	$(1,4,3,3)$
$(-1,1,2,2)$	StrongPosMinus	$(-3,-2,1,1)$	$=$	$(1,4,3,3)$
$(-1,1,2,2)$	StrongNegMinus	$(-3,-2,1,1)$	$=$	$(2,3,3,3)$
$(-1,1,2,2)$	NonMul	$(-3,-2,1,1)$	$=$	$(-3,3,9,9)$
$(-1,1,2,2)$	StrongPosMul	$(-3,-2,1,1)$	$=$	$(-2,3,1,9)$
$(-1,1,2,2)$	StrongNegMul	$(-3,-2,1,1)$	$=$	$(-3,2,9,1)$
$(-1,1,2,2)$	NonDiv	$(-3,-2,1,1)$	$=$	$(-0.5,0.5,2.5,2.5)$
$(-1,1,2,2)$	StrongPosDiv	$(-3,-2,1,1)$	$=$	$(-0.5,0.33,2.5,0.4167)$
$(-1,1,2,2)$	StrongNegDiv	$(-3,-2,1,1)$	$=$	$(-0.33,0.5,0.4167,2.5)$

$(2,3,1,1)$	NonPlus	$(-1,1,2,2)$	$=$	$(1,4,3,3)$
$(2,3,1,1)$	StrongPosPlus	$(-1,1,2,2)$	$=$	$(1,4,3,3)$
$(2,3,1,1)$	StrongNegPlus	$(-1,1,2,2)$	$=$	$(3,2,3,3)$
$(2,3,1,1)$	NonMinus	$(-1,1,2,2)$	$=$	$(1,4,3,3)$
$(2,3,1,1)$	StrongPosMinus	$(-1,1,2,2)$	$=$	$(1,4,3,3)$
$(2,3,1,1)$	StrongNegMinus	$(-1,1,2,2)$	$=$	$(3,2,3,3)$
$(2,3,1,1)$	NonMul	$(-1,1,2,2)$	$=$	$(-3,3,9,9)$
$(2,3,1,1)$	StrongPosMul	$(-1,1,2,2)$	$=$	$(-2,3,1,9)$
$(2,3,1,1)$	StrongNegMul	$(-1,1,2,2)$	$=$	$(-3,2,9,1)$
$(2,3,1,1)$	NonDiv	$(-1,1,2,2)$	$=$	Division by zero!
$(2,3,1,1)$	StrongPosDiv	$(-1,1,2,2)$	$=$	$(-2,3,-1.67,-1.67)$
$(2,3,1,1)$	StrongNegDiv	$(-1,1,2,2)$	$=$	$(-3,2,-1.67,-1.67)$

$(-3,-2,1,1)$	NonPlus	$(-1,1,2,2)$	$=$	$(-4,-1,3,3)$
$(-3,-2,1,1)$	StrongPosPlus	$(-1,1,2,2)$	$=$	$(-4,-1,3,3)$
$(-3,-2,1,1)$	StrongNegPlus	$(-1,1,2,2)$	$=$	$(-2,-3,3,3)$
$(-3,-2,1,1)$	NonMinus	$(-1,1,2,2)$	$=$	$(-4,-1,3,3)$
$(-3,-2,1,1)$	StrongPosMinus	$(-1,1,2,2)$	$=$	$(-4,-1,3,3)$
$(-3,-2,1,1)$	StrongNegMinus	$(-1,1,2,2)$	$=$	$(-2,-3,3,3)$
$(-3,-2,1,1)$	NonMul	$(-1,1,2,2)$	$=$	$(-3,3,9,9)$
$(-3,-2,1,1)$	StrongPosMul	$(-1,1,2,2)$	$=$	$(-2,3,1,9)$
$(-3,-2,1,1)$	StrongNegMul	$(-1,1,2,2)$	$=$	$(-3,2,9,1)$
$(-3,-2,1,1)$	NonDiv	$(-1,1,2,2)$	$=$	Division by zero!
$(-3,-2,1,1)$	StrongPosDiv	$(-1,1,2,2)$	$=$	$(-2,3,-1.67,-1.67)$
$(-3,-2,1,1)$	StrongNegDiv	$(-1,1,2,2)$	$=$	$(-3,2,-1.67,-1.67)$

Appendix B

Historical Survey of Analogue Circuit Design Tools

To enhance the productivity of circuit design, the synthesis of electronic circuits by computer has been part of vigorous research for many years. Significant progress has been made in the automation of digital circuit design.

Before existing approaches, methods, and tools for analogue circuit design are discussed the impractical use of digital circuit design tools for analogue circuit design is discussed in this Chapter.

B.1 Impractical Use of Digital Circuit Design Tools for Analogue Circuit Design

Digital implementation methodology is the key of the 90's. But even the most digital-based devices have to interface with the rest of the world, and analogue circuits provide that gateway. Analogue interfaces in digital systems are needed for:

> transmission media, audio I/O, physical sensors and actuators, imagers and displays, storage media, etc.

Typical analogue circuits which are found continuously in interfacing the real world with VLSI (Very Large System Integration) digital systems are:

Operational Amplifiers, Instrumentation Amplifiers, Voltage References, Sample/Hold Amplifiers, A/D Converters, Analogue Multiplexers, Phase-Locked Loops, Analogue Multiplexers, Comparators, Video/Wideband Amplifiers, Power Amplifiers

Tools are quite common for digital circuit design support and automation which enable the design, of very complex circuits like CPU's, ALU's, memories, etc. Using these digital-based design tools is not possible for analogue circuit design. The differences between analogue circuit design and digital circuit design does not allow this. These are:

- The size of circuits which has to be tackled during a design. Analogue circuits tend to have much fewer transistors than digital circuits.

- The complete range of an analogue component is usually exploited but digital components working range is limited basically to two states.

- In digital circuit design a hierarchical level of design abstraction is developed which the digital circuit synthesis tools follow. In analogue circuit design there is not a hierarchical way of design. Nevertheless, hierarchy is an important issue in circuit synthesis for large complex circuits. It is not possible to attack it in a transistor-by-transistor fashion.

- The consideration of process constraints during the design. At a high level design of digital circuits the process which is used to produce ICs is stated in a simpler form (e.g. driver capabilities, speed, limits of the used transistor technology). In analogue circuits fabrication, for example, there are limits in circuit precision and limits in availability of nominal component values.

- The behavioural description of digital circuits can be done with sets of input to output bit patterns. In analogue the specifications may take the form of

a set of performance parameters that must be met, such as gain bandwidth, noise, phase margin, etc.

Nowadays more and more integrated circuits have analogue and digital circuits integrated in one chip, the mixed analogue digital circuits. The design of the analogue circuit is the most time consuming part of the overall mixed analogue digital circuit design. This bottleneck of the circuit development is resolved through the development of new tools and approaches for the design automation of analogue circuits and their interfacing to digital circuits.

B.2 Present Status of Analogue Circuit Design Tools

Analogue design automation tools can be classified in various ways. Toumazou ([Toumazou and Makris, 1995]) categorized different analogue design automation approaches as: layout based, optimization based, and knowledge based. Rutenbar ([Rutenbar, 1993]) classified them only into: simulation based and equation based. Fujita, Mori, and Mitsumoto ([Fujita et al., 1986]) used an algorithmic based and knowledge based classification. One could categorize design tools based on approaches like genetic algorithms, statistics, constraint-driven, top-down, bottom-up, etc. All of them might be useful depending on the point of view and emphasis of the researcher. Existing tools often combine approaches and are therefore cannot be classified in one of the given classifications exclusively. What is possible is a classification of the tools by taking into account their main approach. The classical classification of bottom-up and top-down is taken in this thesis. This categorization is very general, has a clear boundary, and can contain all other approaches mentioned before, and include new approaches like genetic algorithms based approach which is not mentioned in either of the categorizations above. It is possible to subdivide the top-down approach into three main sub-classes:

- **Knowledge-Based:** Domain knowledge represented by heuristics is used to solve the design problem. The heuristics are found by knowledge acquisition of the circuit design engineers solving design problems and intuition working with the systems and improving them.

- **Hardware-Description-Language-Based:** Synthesis of the circuits by refinement of the hardware description language until a correspondent circuit description of a circuit library is found. This is a high-level approach for designing circuits. There are some heuristics helping the system recovering dead ends during the search and refinement algorithms.

- **Algorithmic-Based:** There are algorithms known which applied to the synthesis problem generate the circuit topology and size the circuit components. However, most tools are limited to perform optimization.

After classification and discussion of existing analogue circuit synthesis tools, an overview of the current research into design activity in general is given. This chapter discusses the uncertainty involved during the design of systems, and argues that the absence of certainty makes a *qualitative* and *fuzzy* based design approach necessary.

B.3 Bottom-Up Approaches

The bottom-up approach is used most frequently in the design of semi-custom integrated circuits. A semi-custom bottom-up approach is controlled to a large extent by the layout. Approaches to implement analogue circuits on semi-custom arrays can be found in [Duchene et al., 1993] and [Mehranfar, 1991]. Semi-custom systems are systems which are based on standard layout library cells of a specific integrated circuit technology.

Definition B.99 (*Circuit Cell*): A circuit cell is a typical functional block of an analogue circuit with several tens of transistors.

Analogue circuit modules are synthesized by module generators which are based on parameterized custom circuit cells for common analogue functions. Semi-custom system have no predefined component groups or circuit structure (e.g. gate arrays) which must be used for all cells and applications. The cell based systems provide a library of functional blocks each of which has been optimally designed by design experts using whatever components were applicable to design that particular function. The high level specifications are met by connecting the functional blocks. Often there is no explicit high level. The advantage of this approach is the fact that the connection of abstract functional blocks often can be used as an adequate specification. The disadvantage is the strong dependency on the library the semi-custom circuit relays on.

The cell-level analogue circuit synthesis of the Carnegie Mellon University ([Maulik et al., 1992],[Maulik and Carley, 1991]). uses a mixed-integer nonlinear programming approach which allows them simultaneous topology selection and parameter selection. Topology choices are represented as binary integer variables and design variables are represented by continuous variables. To solve the mixed-integer nonlinear programming problem they use the branch and bound approach.

VITTOLD, described in [Degrauwe et al., 1989], is able to generate the layout of biquad and leapfrog SC filters starting form filter specifications which include amplitude and phase response. The tool performs dynamic range optimization, Monte-Carlo simulations for yield prediction, and is interfaced with a switched-capacitor simulator. The approach of this tool is restricted to filter design.

FPAD [Fares and Bozena, 1995] is a fuzzy nonlinear programming approach for the design of cell-level analogue circuits like operational amplifiers. FPAD searches for the optimal parameter values (e.g. length and width of CMOS transistors layout structures) starting with input specifications for the specific analogue circuit cell to be designed. Fuzzy set theory is used to measure the degree of fulfillment of the objectives and constraints, and hence provides a measure of the design quality. Trade-offs are handled by manipulating the shape of the membership functions that

reflect the fulfillment or violation of the performance specifications. The mathematical formulation of the design problem is fuzzified to support real-world terms like: high, good, small, acceptable, etc. There are several problems to be solved. First this approach is based on simplified analytic models used in the first sizing procedure, and these are hard to get. The exact nature of simplified models are highly dependent on the individual designer. Second this approach depends highly on the membership function for the fuzzy objective function, which is defined by the individual designer. But most designers do not know what membership functions are.

In [Horrocks and Khalifa, 1994], [Horrocks and Khalifa, 1995], [Koza et al., 1996], and [Horrocks and Arslan, 1995] genetic algorithms are used to find filter topologies using predefined filter topology fragments. The search space is limited by allowing only sensible combinations of the filter topology fragments. At each generation new filter topologies are generated by mutation and sexual reproduction using the filter topologies of the old generation. First it is not easy to ensure correct filter topologies after a mutation. Second at each generation the filter topology with the highest fitness is selected for further reproduction. This fitness is calculated by simulating the filter circuits. The time needed during simulation of the filter circuit constitutes a problem for using genetic algorithms for circuit design, at the component abstraction level.

B.4 Top-Down Approaches

The top-down approach tries to break down the high level specifications to block-level description, from block-level description to sub-block-level description, from sub-block-level description to circuit schematic, and from circuit schematic to integrated mask structures. Most existing synthesis tools perform the reduction in two steps. First they reduce the high level specification to a circuit schematic and second transform the circuit schematic to a structure from which, the integrated

circuit mask can be produced. Top-down approaches are: hardware description language-based, algorithmic-based systems and knowledge-based systems.

B.4.1 Hardware-Description-Language(HDL)-Based Systems

The ability to describe, simulate, and synthesize circuits as well as complete systems from a HDL is an option that many engineers are choosing today for their most complex system designs [Fox, 1993]. Hardware description languages satisfy a number of needs in the design process. Firstly, a HDL allows description of the structure of a design, that is how it is decomposed into sub-designs, and how those sub-designs are interconnected. Secondly, it allows the specification of the function of designs using familiar programming language forms. Thirdly, as a result, it allows a design to be simulated before being manufactured, so that designers can quickly compare alternatives and test for correctness without the delay and expense of hardware prototyping. Most significant is the possibility for the designer to describe its behaviours by building a procedural model (functional block) of the design. It is not necessary to know the internal structure of the circuit block. After testing the correctness of the model by simulation synthesis, tools generate concrete circuits. This synthesis from a behavioural specification of a design is often called *silicon compilation.*

At present there are basically only synthesis tools based on the Very High Speed Integrated Circuits (VHSIC) Hardware Description Language called VHDL (VHDL standard 1993: [VHDL Standard 93, 1994]). VHDL was developed to describe very complex digital circuits. Since a description with VHDL can not necessarily be completely synthesized [Camposano et al., 1991] quite a number of tools arose sometimes coupled with design frameworks, e.g. BECOME [Wei, 1988], CADDY [Camposano, 1989], Siemens's Synthesis System [Scheichenzuber, 1990], SIS [Sentovich et al., 1992], SYNOPSIS [Kurup and Abbasi, 1995].

Analog hardware description languages have been also emerging not only because of the difficulty of synthesizing but also because of the need to combine analogue and digital circuits. A future standard for analogue hardware description language (AHDL or VHDL-A or VHDL 1076.1) is the extension of the VHDL to continuous time systems. The AHDL should be suitable for the description and simulation of mixed, analogue-digital, systems. VHDL-A supports behaviour specification of an analogue system by a set of linear/nonlinear differential/algebraic equations and/or by a sequence of assignments. It is likely that 1999 the 1076.1 Language Reference Manual (LRM) will become the IEEE 1076.1 standard [VHDL-A Standard 99, 1999].

A framework which supports the design of mixed circuits is the system called KANDIS [Grimm et al., 1995]. It allows the designer to specify the design with a language called VHDL-Hybrid. These languages are used to generate RTL-VHDL, net-list, VHDL, and VHDL-A code. RTL-VHDL can directly be used for synthesizing digital circuits. The net-list is used to develop a layout of the circuit by hand. VHDL and VHDL-A code is used to simulate the circuit. The automatic synthesis of the VHDL-A to a real analogue circuit is not solved yet.

B.4.2 Algorithmic-Based Systems

Algorithmic-based systems use algorithms to meet the user-defined specifications. Since it is hard to find algorithms for circuit synthesis, most of the synthesis tools are limited to perform optimization which is often hard too. Very often algorithmic-based systems are classified into optimization-based systems. Optimization tools use a fixed circuit structure to adjust the circuit parameter until they meet the specifications. The first attempts towards design automation were systems like DELIGHT.SPICE [Nye et al., 1988], ECSTACY [Shyu and Sangiovanni-Vincentelli, 1988], and ADOPT [Lai et al., 1988]. Onodera, Kanbara, and Tamaru [Onodera et al., 1990] developed an operational amplifier

compiler with performance optimization. Intelligent optimization systems use different kind of optimization algorithms and select the best suited algorithm to compute the best result it can achieve. Nevertheless, if no result can be computed they output explanations for the user and suggest further design actions.

DELIGHT.SPICE [Nye et al., 1988] is a software package dedicated to optimize component values for a fixed circuit topology. The system optimization-based approach uses various optimization algorithms combined with circuit simulations techniques to produce an acceptable parameterized circuit. DELIGHT.SPICE combines the optimization-based CAD system DELIGHT with the circuit analysis program SPICE [Nagel, 1973]. For optimization a circuit scheme must be known and the optimization problem formulation must be defined that a powerful optimization algorithm can optimize the circuit. After the system DELIGHT computes the circuit parameters the simulator SPICE checks the result. If the result meets the specification, the final parameter values are reported. If it does not meet the result it performs a sensitivity analysis to give the system DELIGHT information about the parameters which are most likely to be changed to achieve the specifications.

Some of the Computational Intelligence-based optimization tools are based on simulated annealing [Gielen et al., 1990] or genetic algorithms. Genetic algorithms justified their usefulness in the area of optimization, as showed in [Horrocks and Spittle, 1993], by optimizing the component values of filters. A symbolic-based sizing approach as Sommer [Sommer and Henning, 1995], has been suggested. Chen and Yang [Chen and Yang, 1995] use a statistical design approach based on non-parametric performance macro-modelling (STYLE).

B.4.3 Knowledge-Based Systems

Another approach to treat analogue circuit design is an Artificial Intelligence approach using expert systems. Expert systems very often combine the algorithmic approach with an heuristic problem solving approach. The heuristics are used to

find a solution for the complicated problem of analogue circuit design whenever an algorithmic approach fails to find a solution. Knowledge is represented as facts, design rules, actions, and their relationships about a subject area. The domain specific design knowledge integrated into these systems permits novice designers to specify simply their design requirements and by designing analogue circuits become reasonably good designers in an easy-to-learn way. The significant parts of an expert system are knowledge base, inference engine, and the control component. The following systems represent presently well-known existing analogue circuit synthesis tools which are described in detail:

1. **IDAC** [Degrauwe et al., 1987] is a commercial interactive synthesis system, the architecture shown in Fig.B.1:

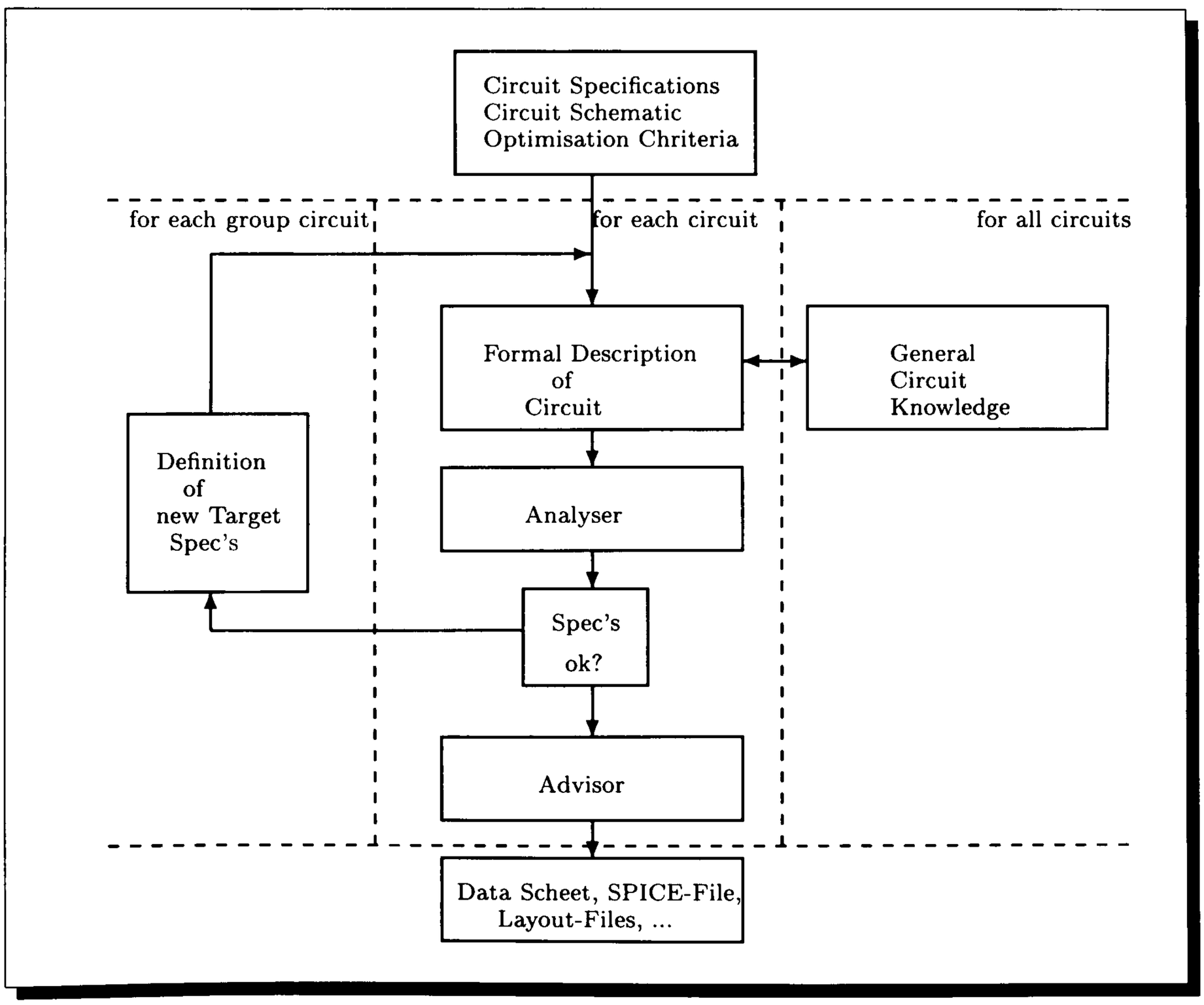


Figure B.1: System-Architecture of IDAC

It is able to design transconductance amplifiers, operational amplifiers (op amps), low-noise BiCMOS amplifiers, voltage and current references, quartz oscillators, comparators, and oversampled A/D-converters including their digital decimation filter. Each of the above mentioned circuits has its own synthesis strategy guided by the control component and is implemented as a fixed library topology and a formal description. Hence, each individual circuit synthesis can be seen as an individual program for designing specific circuit blocks. IDAC makes use of three types of knowledge:

(a) knowledge specific to the schematic (e.g. existence of different op amps),

(b) general circuit knowledge (e.g. how to size cascade devices), and

(c) knowledge common for a family of circuits (e.g. how to stabilize circuits).

All the design knowledge is stored as circuit topology knowledge and analytical synthesis equations. The selection of the circuit topology and information about the technology parameters must be provided by the user. To guide the tool during the optimization phase, design options can additionally be specified. The synthesis process is limited in sizing the devices using an algorithmic design strategy consisting of synthesis equations. These analytical equations are sequentially ordered to form a single-pass algorithm which involves no backtracking. The result of this process is a schematic with sized devices. If the design is deemed close enough to the specifications, a second numerical optimization phase is performed to tune the final circuit. After the sizing phase of each device, the program will use a built-in analyzer to verify the feasibility of the design. If the analyzer detects that some specifications are not satisfied, then a new set of target specifications is defined and the formal description is re-executed. This is repeated until all specifications are satisfied. At the end of the synthesis phase IDAC gives the user additional information, so called warnings when e.g. leakage currents have to be taken into consideration, high-

frequency effects transistors. The output of IDAC is a detailed input listing, a comparative table of the most important characteristics of various schematics. a complete data sheet, a SPICE2 [Nagel, 1975] input file, detailed behaviour descriptions, various circuit response descriptions, and an input file for the layout program ILAC.

2. **BLADES** [El-Turky and Perry, 1989] combines, unlike the previous design synthesis tool (IDAC 1), the formal knowledge with intuitive knowledge (heuristics). BLADES architecture (Fig.B.2) consists mainly of five large sub-systems (expert design manager, sub-circuit design expert, knowledge base, test generator, and design consultant) each of which is specialized in one or more design aspects.

   The first step in the synthesis process is that the expert design manager tries to determine a circuit topology from the circuit specifications given by the user. The circuit topologies are stored in the knowledge base described in a hard-ware description language. BLADES uses three different hardware description languages to describe the circuits in different level of abstraction and to code the design rules. After finding the high level circuit topology the expert design manager determines from the input specifications the functional requirements of each sub-circuit.

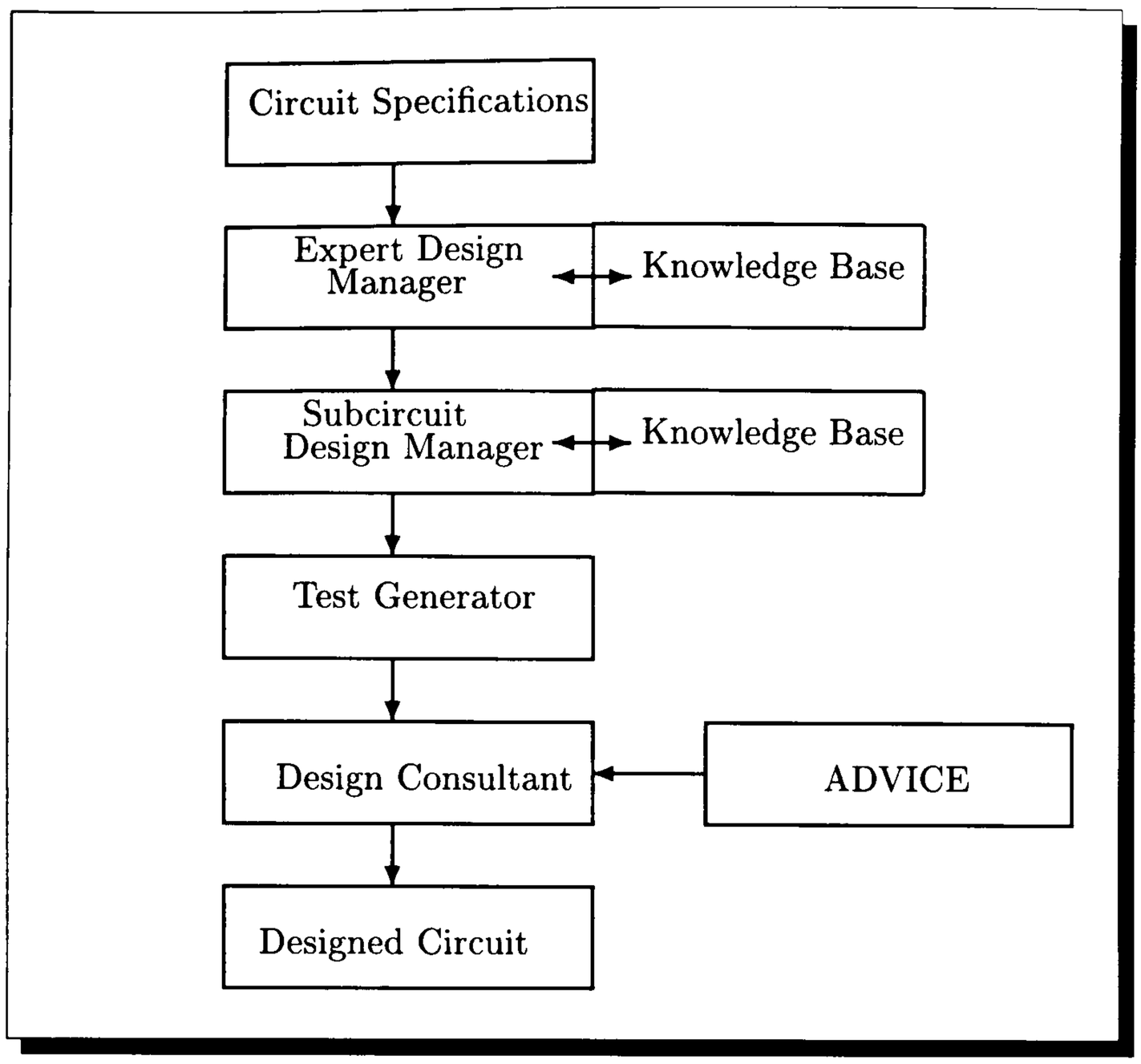


Figure B.2: System-Architecture of BLADES

The sub-circuit expert has its own private knowledge base. Each expert is specialized in a single class of sub-circuits. These sub-circuits are represented in the knowledge base by its topology and a set of rules to calculate the component values. Once a sub-circuit has been selected the sub-circuit expert is performing the exact calculation of the component values. After the synthesis of the circuit is complete the test generator is used to check the correctness of a circuit. It allows the designer to review the reasoning and alleviate potential design problems. In BLADES it is assumed that the system understands the design objectives completely. Thus its ability to generate consistent test procedures for a particular design is guaranteed. The correctness check is done by the design consultant which analyzes the design circuit using the simulator ADVICE[1].

[1] ADVICE is used and developed at AT&T Bell Laboratories.

3. **OASYS**'s [Harjani et al., 1989] key concept is based on the assumption that the problem of transferring high level description of a circuit to an analogue circuit structure can be solved by decomposition of hierarchical representation of circuits interactively. The hierarchical decomposition permits the tool to partition the design problem into a number of smaller, more manageable, and more independent synthesis problems. The system architecture of OASYS (Fig.B.3) consists of a topology selector, sub-circuit generator through plans and plan-fixers, and an optimizer.

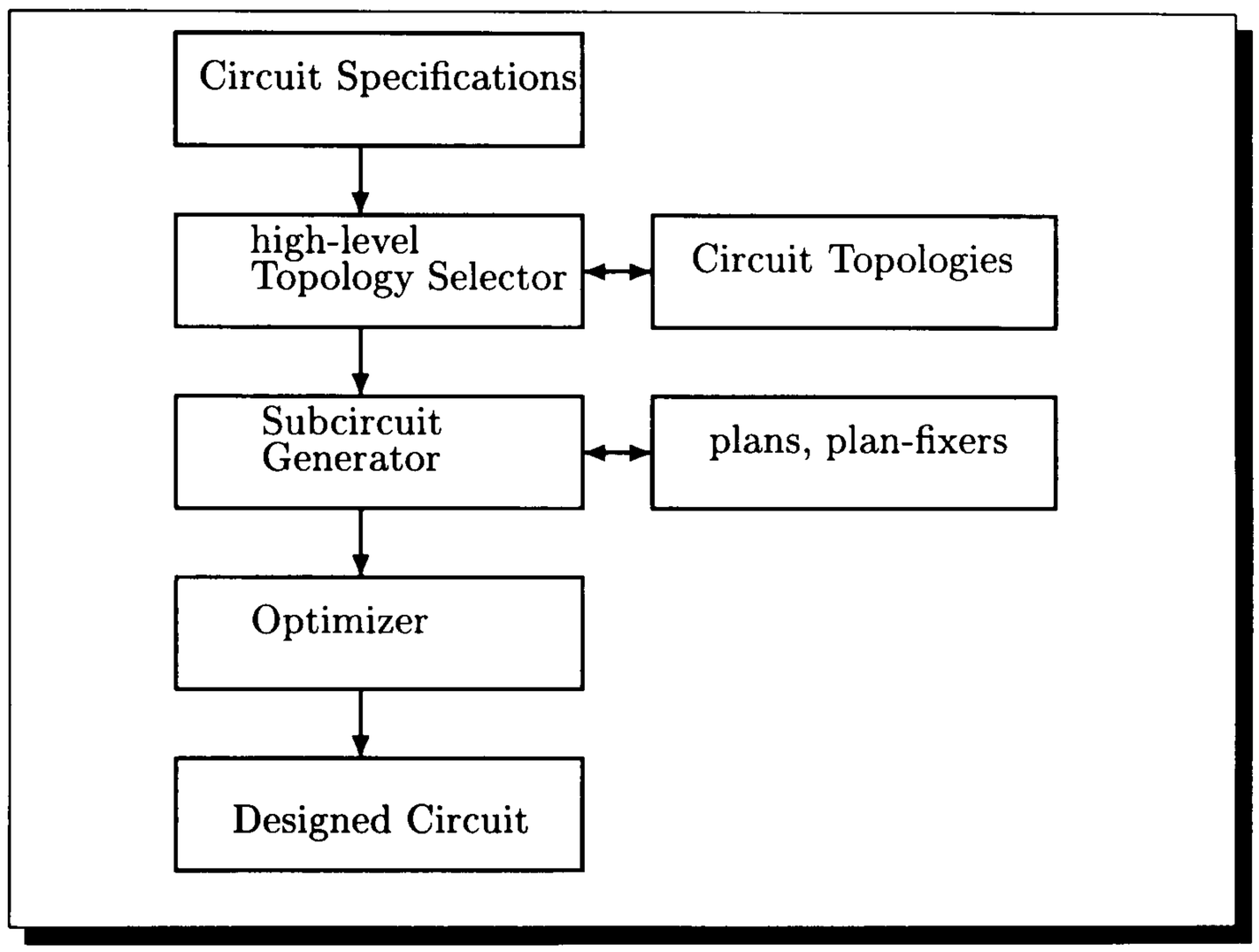


Figure B.3: System-Architecture of OASYS

At the highest level of hierarchy there are fixed circuit topologies, called design styles, which are implicitly represented as statically stored templates of connected sub-blocks. According to the performance specification given by the user, a circuit topology at the highest hierarchy-level is selected through generate-and-test. Once a design style has been selected, OASYS tries to discriminate the chosen topology into sub-blocks until each sub-block is specified. The translation of high level blocks into sub-blocks is done by a planning sys-

tem. For each fixed topology a design plan is associated and executed when the topology is instantiated. A plan in OASYS is a sequence of plan steps. These plan steps can be characterized as:

- **computation:** values have to be computed in order to proceed the design (e.g. voltages or currents have to be known).
- **heuristics:** to make decisions based on expertise and on incomplete knowledge, in order to advance the design state (e.g. choosing good initial values, worst case considerations).
- **refinement:** selection and translation mechanism for a lower level sub-block (e.g. when designing an operational amplifier it has to be designed an differential input pair).

Whenever a plan step finds that it has been unable to meet its goal, control is passed outside the plan to a set of failure handlers, called plan-fixers, which are attached to each plan. The planning system as described above is rather limited since the plan-fixers try to solve the problem by if-then rules, by algorithms, or by changing the order of the sub-problems in the plans. Compared to GPS [Charniak and McDermott, 1985] with the intention to model human problem solving by using the means-ends method, this planner is an extremely domain-dependent planner involving only sequencing of plans. To achieve the variety of design optimizations for a circuit, several design plans for the same circuit topology are implemented as different design styles. OASYS uses a *fixed point iteration style* [2] to optimize the circuit after the structure of the circuit is determined. OASYS is expected to be used by designers to synthesis conventional designs which require the tool simply to produce the best possible design and to explore tradeoffs in the design space. OASYS is a program with about 11000 lines of Franz LISP running under UNIX.

[2] From an initial guess, the plan is used to compute a better value, generate a new improved guess, and iterate again

4. **OPASYN** [Koh et al., 1990] synthesizes a layout of an optimized CMOS operational amplifier taking as input system level specifications, fabrication-dependent technology parameters, and geometric layout rules. The synthesis process can be divided into three tasks which are implemented in a circuit selection module (written in Common Lisp), a parametric circuit optimization module (written in C), and a layout generation module. The architecture (Fig.B.4) consists of these three functional modules and a database which is built up by:

   - a decision tree for topology selection,
   - analytic circuit models for parametric circuit optimization,
   - slicing tree descriptions for floor-planning, and
   - net-list descriptions for routing.

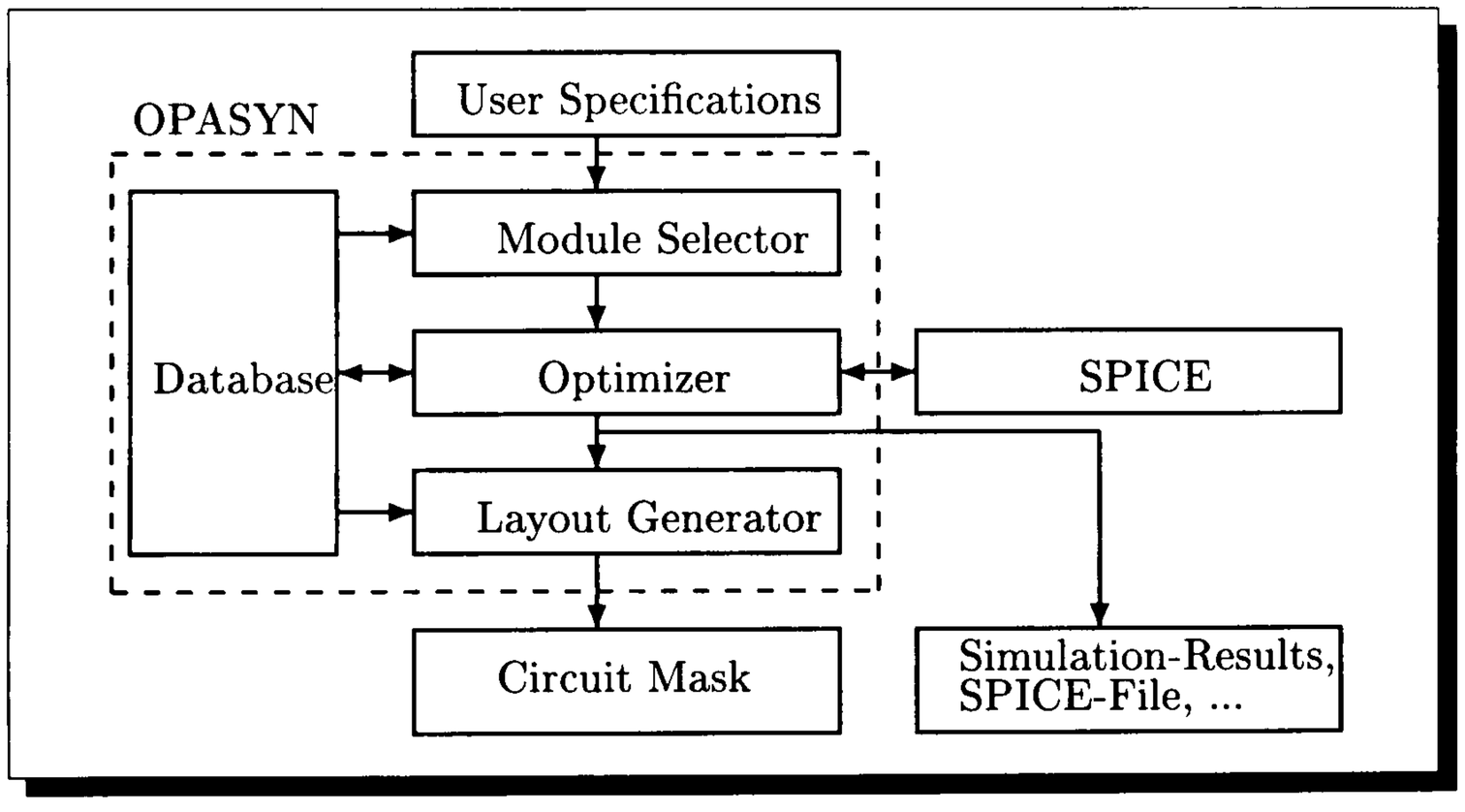


Figure B.4: System-Architecture of OPASYN

The first step in synthesizing an operational amplifier is to search for a promising circuit topology fitting the given design requirements. The circuit topologies are not represented hierarchically, but are flat, fixed topology blocks implemented statically in the database. Therefore to get an entirely "new" topology,

which is not implemented in the database, is quite difficult. It is impossible to reuse previously implemented design knowledge in a new task. Currently five different, widely applicable operational amplifier circuit topologies have been fully incorporated. All these topologies are implemented for CMOS design. It cannot be changed directly to bipolar technology. This has to be implemented additionally. Searching for a suitable topology starts at the root of the implemented decision tree. The circuit-selection is based on heuristic pruning, which checks whether some subtrees can be pruned away (eliminated from further consideration) according to the range of the given specifications. The unpruned leaf nodes are forwarded to the optimization module. The optimization module relies on analytic models which exist for each selected circuit topology. These analytic models typically include:

- net-list description of the circuits,
- declaration of the independent design parameters,
- reasonable upper and lower bounds for the design parameter values, and
- analytic design equations to express dependency of circuit performance on design parameters.

The optimization task is an algorithmic one but substitutes expensive circuit simulation with algebraic evaluation of analytic design equations acquired from expert design knowledge. The verification of the optimized circuit is done with the simulation program SPICE [Nagel, 1973]. When the optimization task is complete and the circuit verification is correct the fully parameterized circuit topology is passed to the layout module which generates mask geometries in a macro cell layout style. Device parameters and design rules for different process technologies are retrieved from a technology library. OPASYN requires a separate program for each circuit schematic, since the synthesis process is done directly from specification to mask geometries.

5. **STAIC** [Harvey et al., 1992] is an interactive synthesis tool that designs CMOS and BiCMOS analogue integrated circuits by accepting structural and performance specifications as input by the user. STAIC (Fig.5) integrates the benefits of analytical modelling, layout dependent modelling, hierarchical circuit representation, and numerical optimization. The circuits are hierarchically represented as:

   categoric blocks: A categoric block declares a set of attributes (variables) that are common to all alternate implementations of a given generic function. (e.g. OTA variables, dc gain, phase margin, etc.)

   specific blocks: A specific block is a particular and unique realization of a generic function (a folded cascade OTA would therefore be classified as a specific block since it is one of many possible OTA realizations).

   styles: A layout style description contains layout directives and model equations relevant to a particular floor-plan and routing scheme.

   devices: At the lowest level of hierarchy, specific blocks are supplanted by devices. Devices have modelling equations relevant to the device. Each device has an associated module generator layout description.

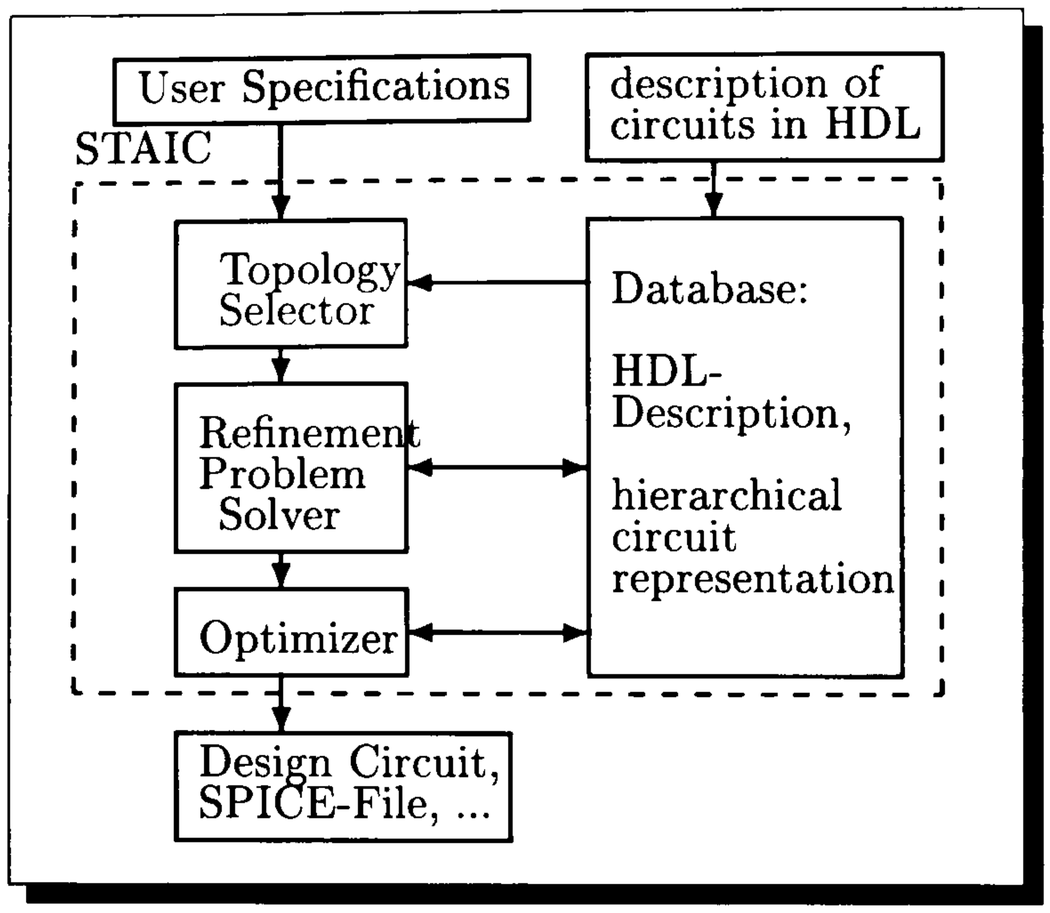


Figure B.5: System-Architecture of STAIC

The hierarchical circuit representation allows the use of a successive refinement technique to cope with design complexity. Before the successive refinement technique is used the synthesis tool selects an initial circuit topology which is represented in a library coded in a hardware description language. The analogue hardware description language allows an expert analogue designer to enter new circuit descriptions without having intimate knowledge of the software internals. Then the successive solution refinement technique exploits the multilevel analytical model descriptions to systematically attain what is likely to be a global optimal solution. In other words this unit dynamically assembles fragmented hierarchical design equations of selected circuit topologies to produce a homogeneous "flat" model description suitable for practical optimization and other numerical methods. Performance specifications take the form of equality and inequality constraints imposed on the attributes of the categoric block of interest. Each constraint has an associated normalized weighting factor, and multiple constraints may be applied to every performance attribute. Design exploration is facilitated by optimizer and scanner modules that interact extensively with a central equation database through a

numeric/symbolic solve unit. After sizing the devices of the circuit which satisfy the constraints it generates a module generator layout description, SPICE [Nagel, 1973] net list, and data sheet as the final output.

Besides the tools already mentioned there are systems which try to automate the design of a switched capacitor filter, using methods and making decisions as a filter designer might (FILSYN [Szentirmai, 1977]). ARIADNE [Swings and Sansen, 1993] has a clear separation between design knowledge and general design procedures. A top-down, constraint-driven design methodology for analogue integrated circuits was developed by a group at the University of California, Berkely ([Malavasi et al., 1993]). Given a set of circuit specifications (circuit characteristics, design rules, technology, and user options), a mapping is made to schematics or to layout. Philip's approach to automated circuit synthesis resulted in the tool MIDAS [Beenker et al., 1993]. MIDAS is an open, knowledge based and technology independent design tool. It is strictly hierarchical, breaking down analogue modules into sub-blocks that may be reused throughout the hierarchy. Similar, the Fraunhofer Institute and TEMIC [Wittmann et al., 1994] developed a top-down hierarchical synthesis tool for analogue circuits but by focusing on the main features of cells and sub-cells.

B.5 Conclusion and Résumé

Common to all systems is that they must have knowledge about structural description of analogue circuits.

IDAC has a basic description of the circuit structure of each circuit it wants to synthesis. BLADES only designs operational amplifiers by rule refinement. OASYS has different circuit topologies implemented for a specific kind of circuits (e.g. Op Amps). At the beginning OASYS does a high level selection by generate and test and does a sub-block refinement by plans. OPASYN is based on analysis and op-

timization of a given topology and STAIC does a successive solution refinement of given topologies.

There is no successful tool which uses component descriptions only and the basic Kirchhoff's laws to construct an analogue circuit. This is because of the immense search space it would arise. A first attempt in this directions are the filter topology generation by Horrocks ([Horrocks and Khalifa, 1994], [Horrocks and Khalifa, 1995], [Horrocks and Arslan, 1995]) and Koza ([Koza et al., 1996]).

At the moment there is no analogue synthesis tool which is able to design a circuit by not knowing what principle circuit structure is underlying. The tools, for example, do not know when to design an operational amplifier or a filter just from the specifications given by the user. Little research has been done into design at a very high prototype level, e.g. block diagram design. It is very important to take into account that the specifications and the models defined by the designer are not known exactly and this must be taken into account during the design process to avoid design mistakes at an early stage. This thesis tackles the problem of analogue circuit design at a very early stage of the design process. Using *qualitative fuzzy simulation* a promising framework (Chapter 9) for an analogue circuit design has been developed.

Appendix C

Pre-Simulated Database

Most of the circuits are examples used by an electronic engineering class in a workshop learning the simulation of analogue circuits.

The following circuits have been pre-simulated:

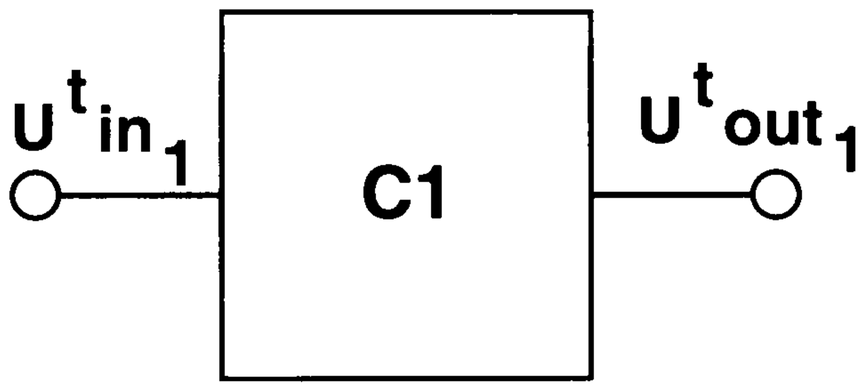
- C1: emitter amplifier with emitter resistor
- C2: differential amplifier
- C3: current mirror
- C4: Wilson current mirror
- C5: operational amplifier
- C6: emitter amplifier
- C7: analog multiplier
- C8: transconductance-c filter

C.1 Circuit C1

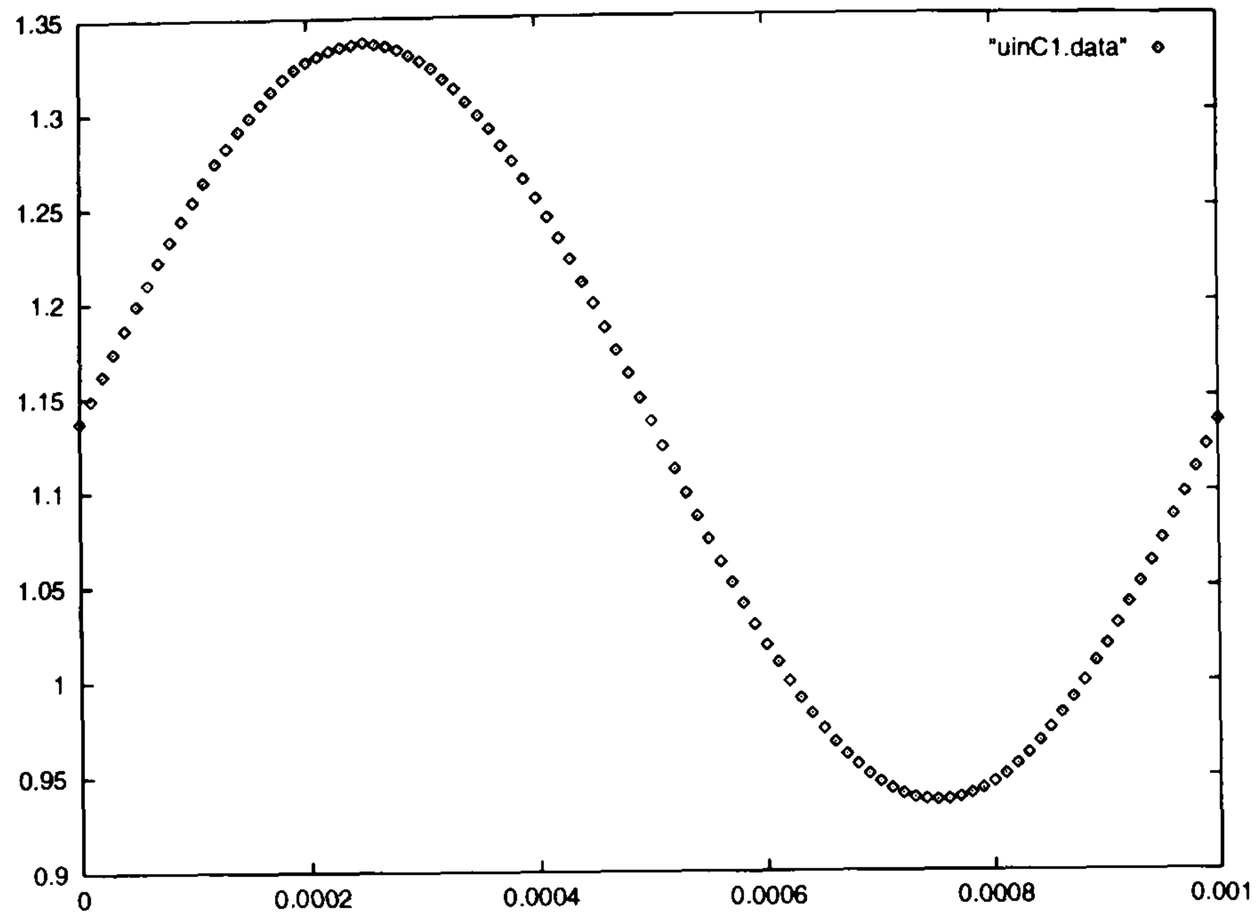
Spice-File:

```
Emitteramplifier with Emitter Resistor
Vcc 3 0 10
Rc 3 2 10k
Re 4 0 1k
Q1 2 1 4 tr1
.model tr1 npn is=10f vaf=100
Vin 1 0 sin 1.13682 .2 1k
.tran 10u 1m
.plot tran V(1,0)
.plot tran V(2,0)
.end
```

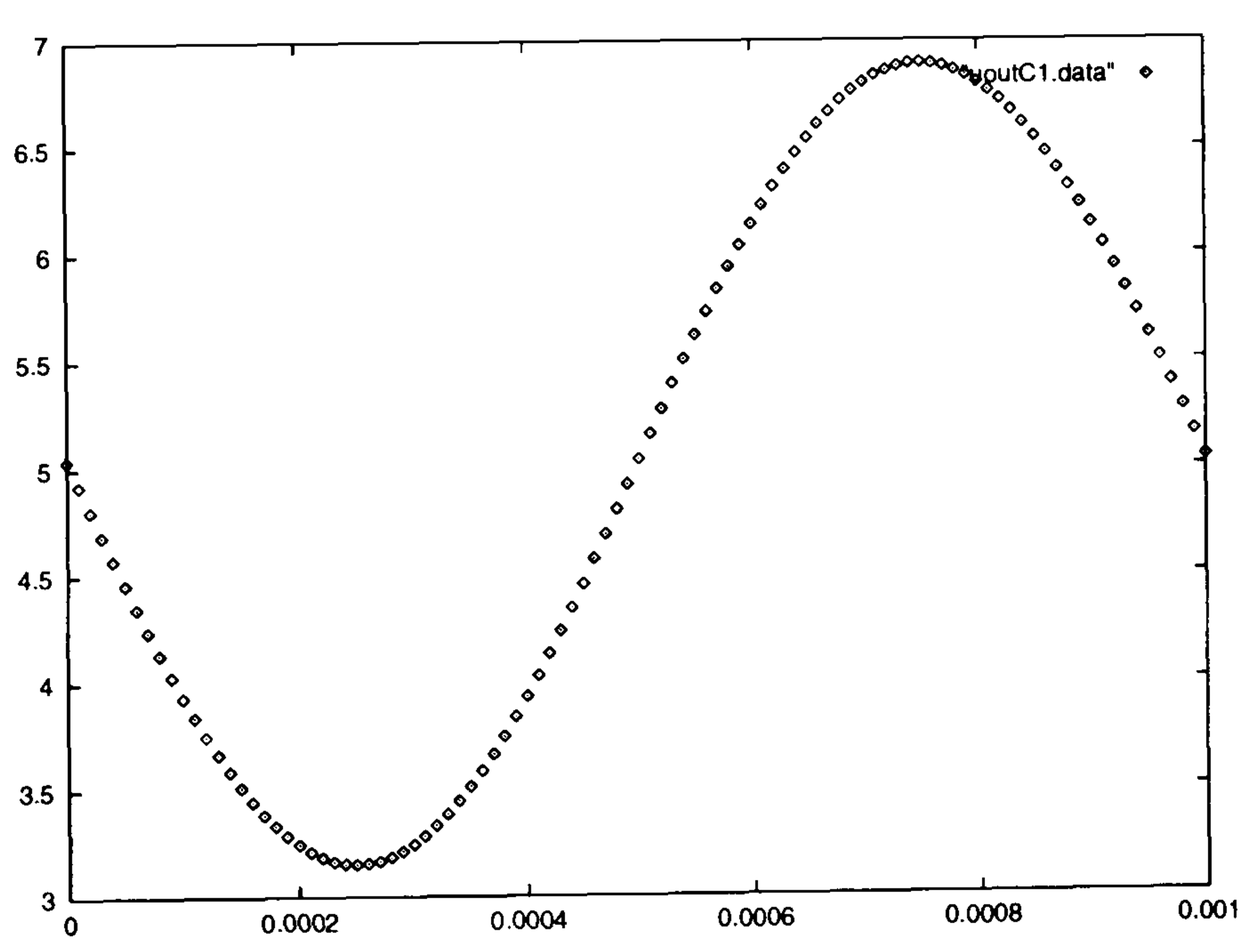
Black-Box Description:



Input Data:



Output Data:



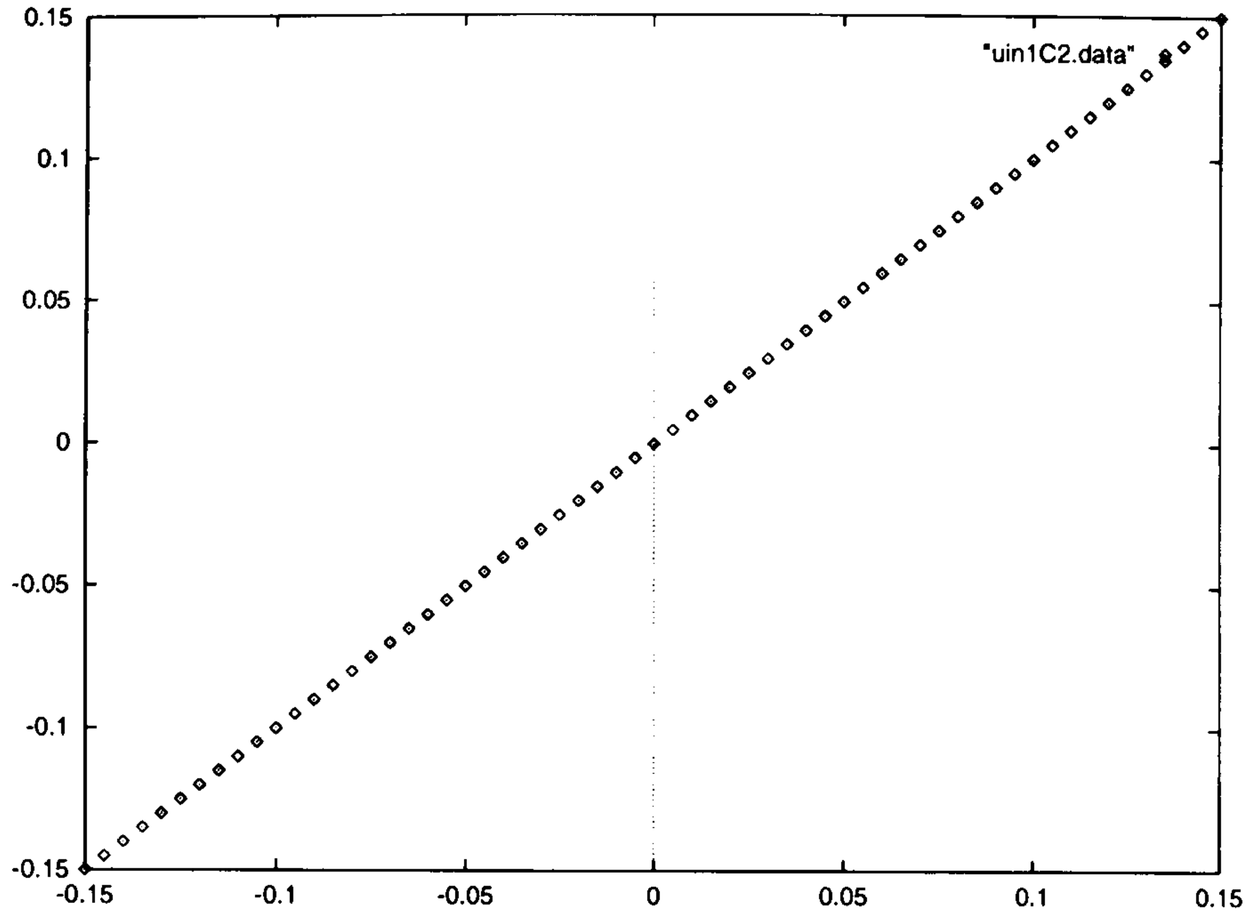
C.2 Circuit C2

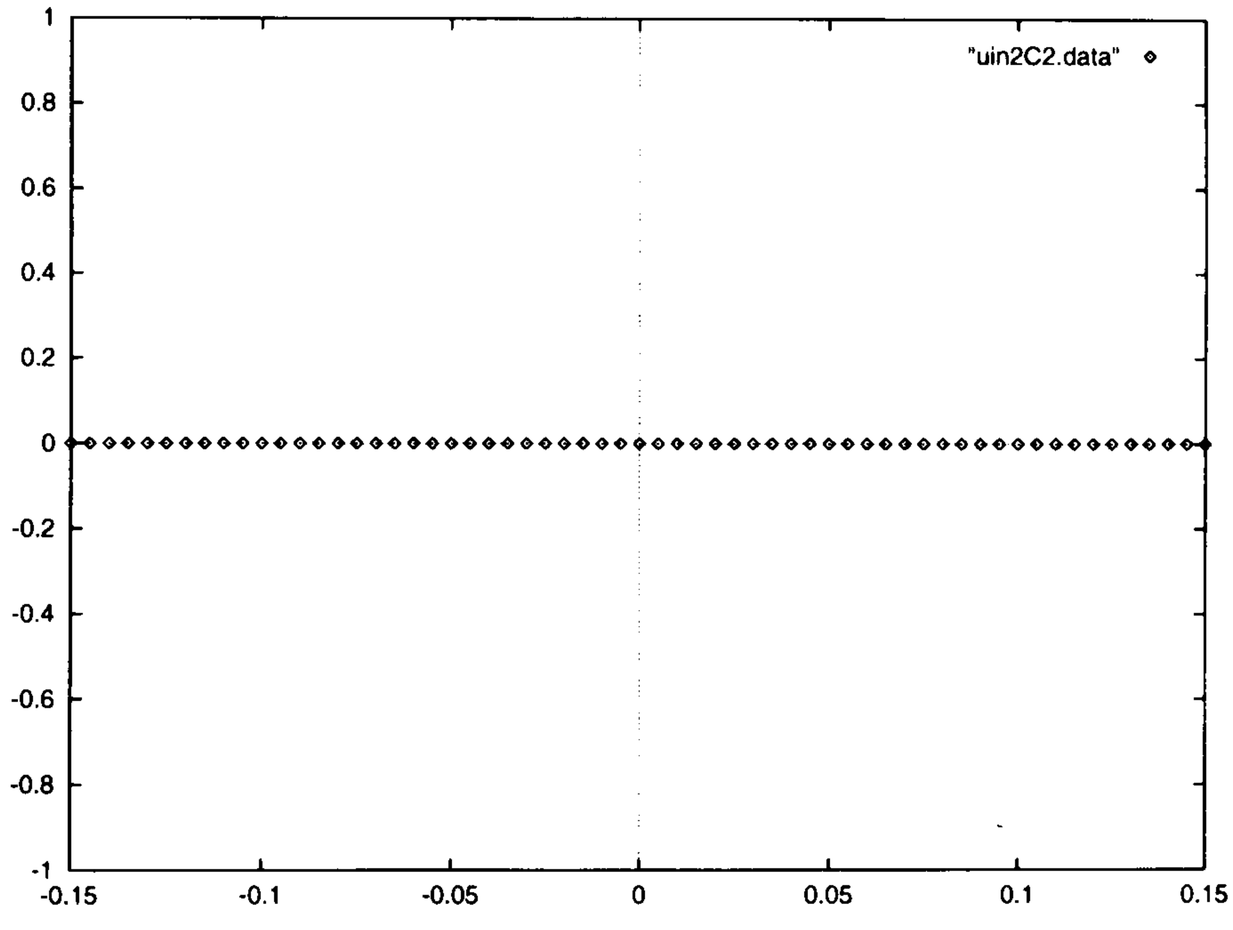
Spice-File:

```
Differential Amplifier

vcc 3 0 10

vee 7 0 -10.7

vdm 1 6

vcm 6 0

rc1 3 2 5k

rc2 3 5 5k

ree 4 7 5k

q1 2 1 4 tr1

q2 5 6 4 tr1

.model tr1 npn is=10f vaf=100

.dc vdm -0.15 0.15 5m

*.dc vcm -12 12 0.1

.op

.tf v(5) vdm

*.plot dc v(5,0)

*.plot dc v(6,0)

.plot dc v(1,6)

.end
```

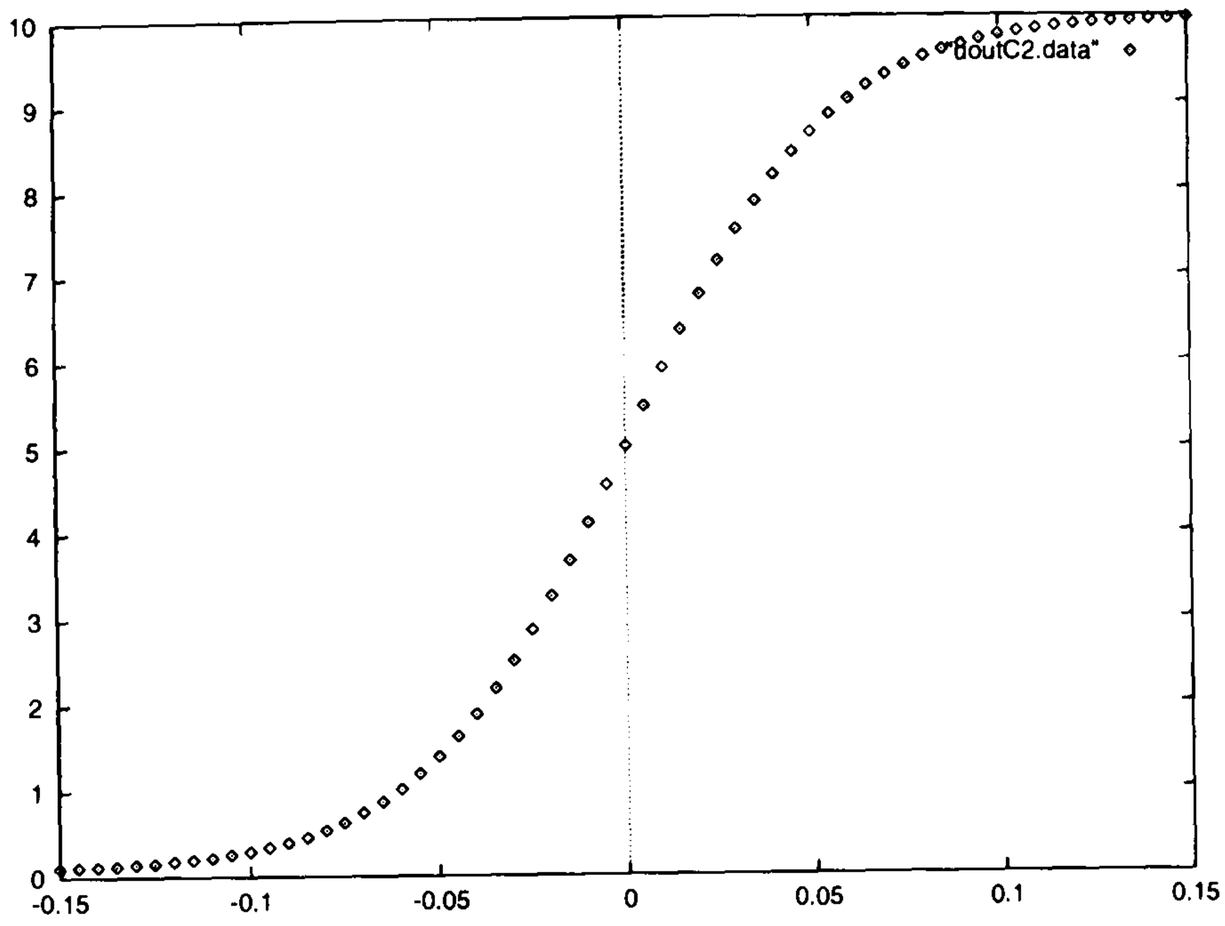
Black-Box Description:

Input Data:


"uin1C2.data"
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.15
-0.1
-0.05
0
0.05
0.1
0.15


"uin2C2.data"
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.15
-0.1
-0.05
0
0.05
0.1
0.15

Output Data:

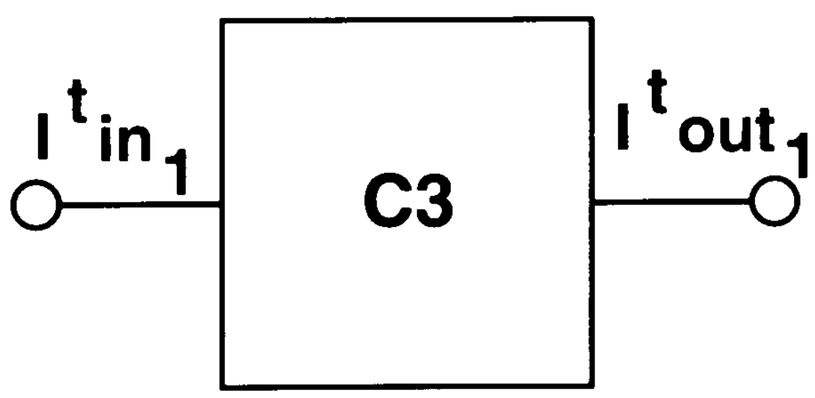


C.3 Circuit C3

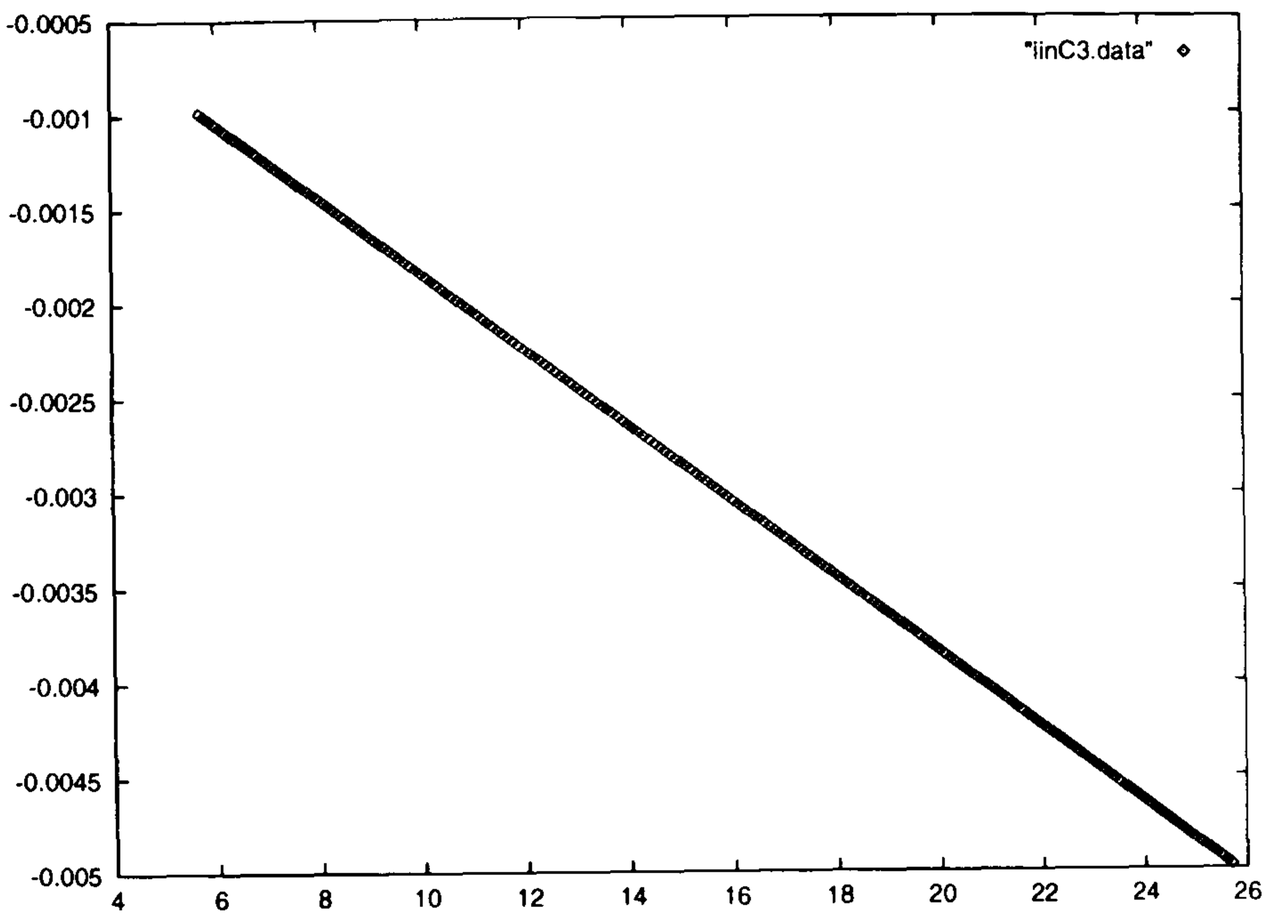
Spice-File:

```
Current Mirror

v1 1 0 15.7

r1 1 2 5k

q1 2 0 0 tr1

q2 3 2 0 tr1

.model tr1 npn is=10f vaf=100

v2 3 0 5

.dc v2 0 30 0.1

.dc v1 5.7 25.7 0.1

.op

.tf i(v2) v1

.plot dc i(v1)

.plot dc i(v2)

.plot dc v(3,0)

.end
```

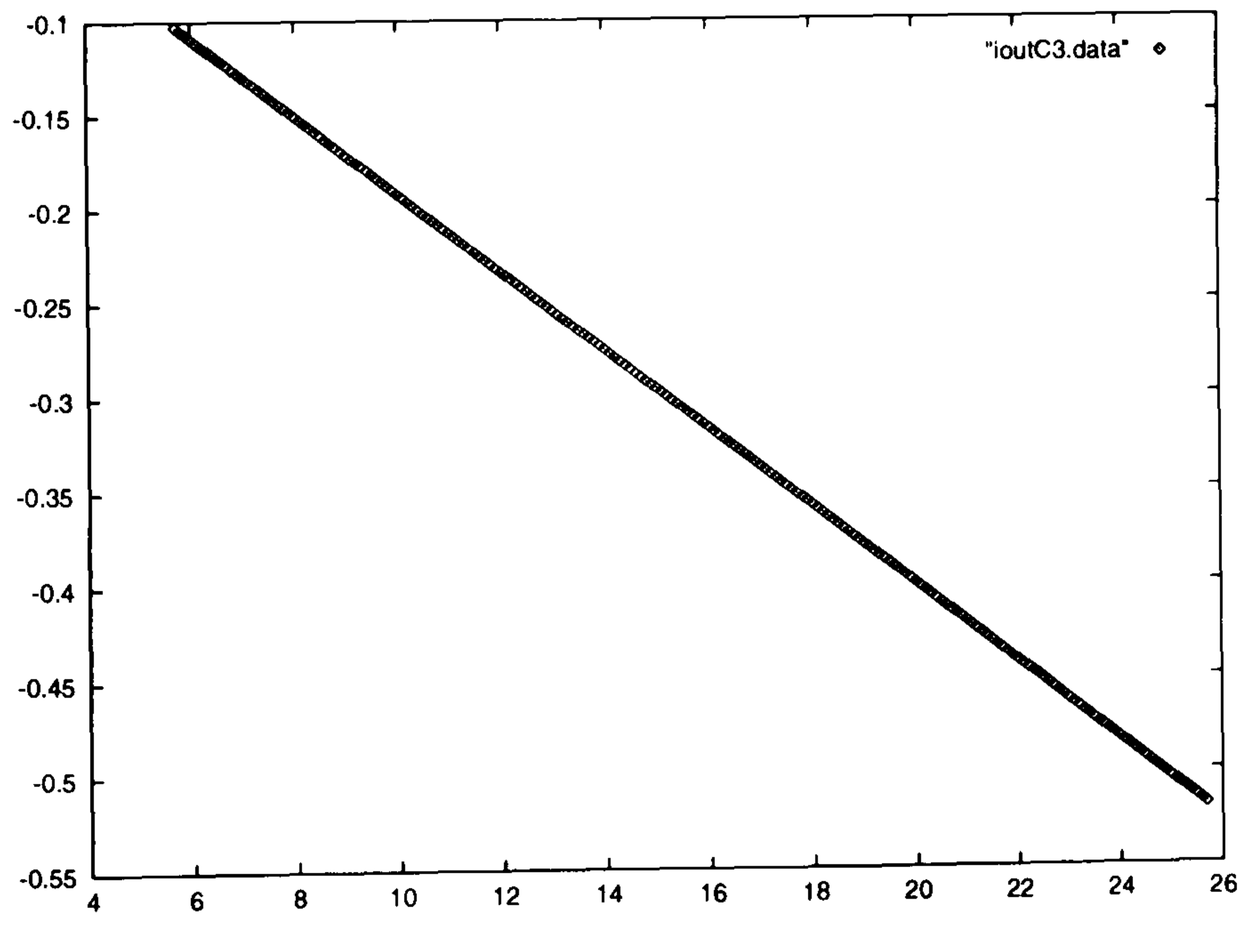
Black-Box Description:



Input Data:



Output Data:

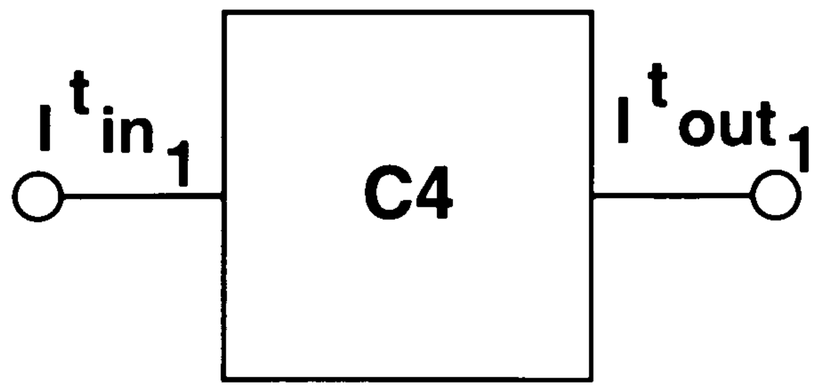


C.4 Circuit C4

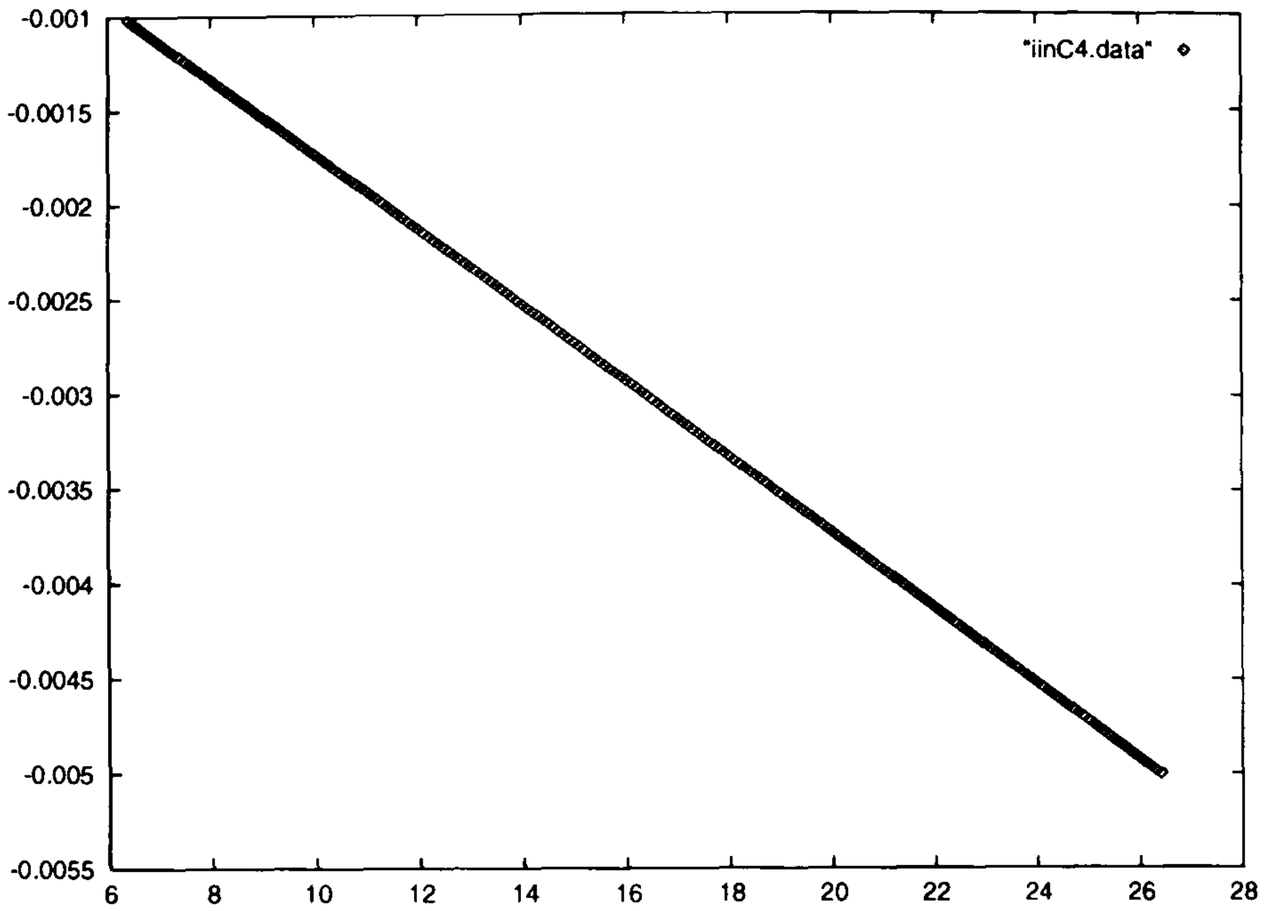
Spice-File:

```
Wilson-Current Mirror

v1 1 0 16.4

r1 1 2 5k

q1 2 3 0 tr1

q2 4 2 3 tr1

q3 3 3 0 tr1

.model tr1 npn is=10f vaf=100

v2 4 0 5

.dc v2 0 30 0.1

.dc v1 6.4 26.4 0.1

.op

.tf i(v2) v1

.plot dc i(v1)

.plot dc i(v2)

.plot dc v(4,0)

.end
```

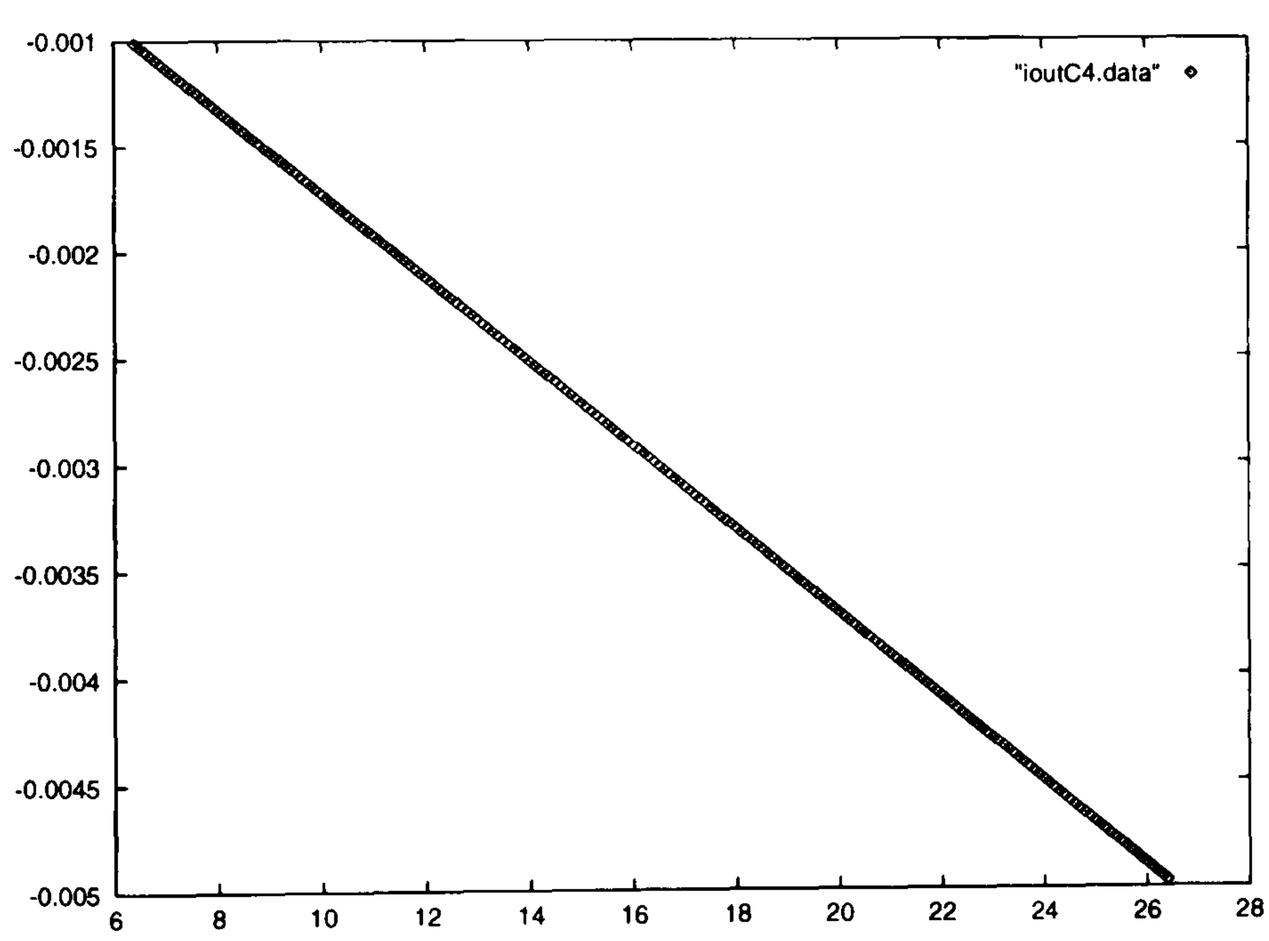
Black-Box Description:



Input Data:



Output Data:

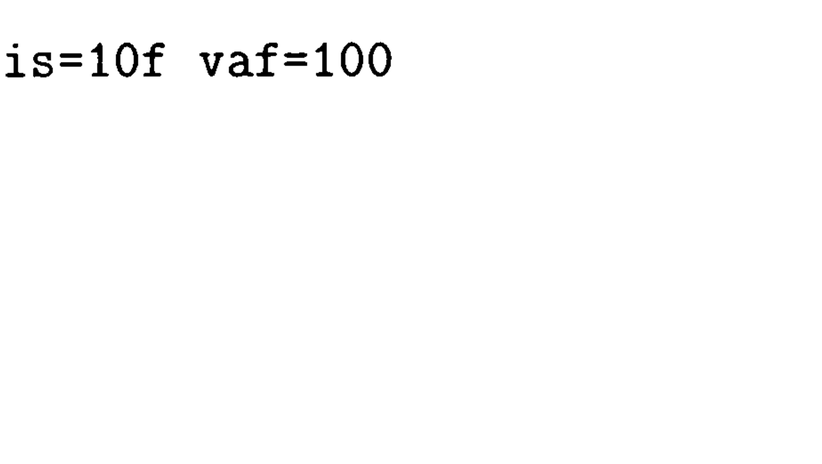


C.5 Circuit C5

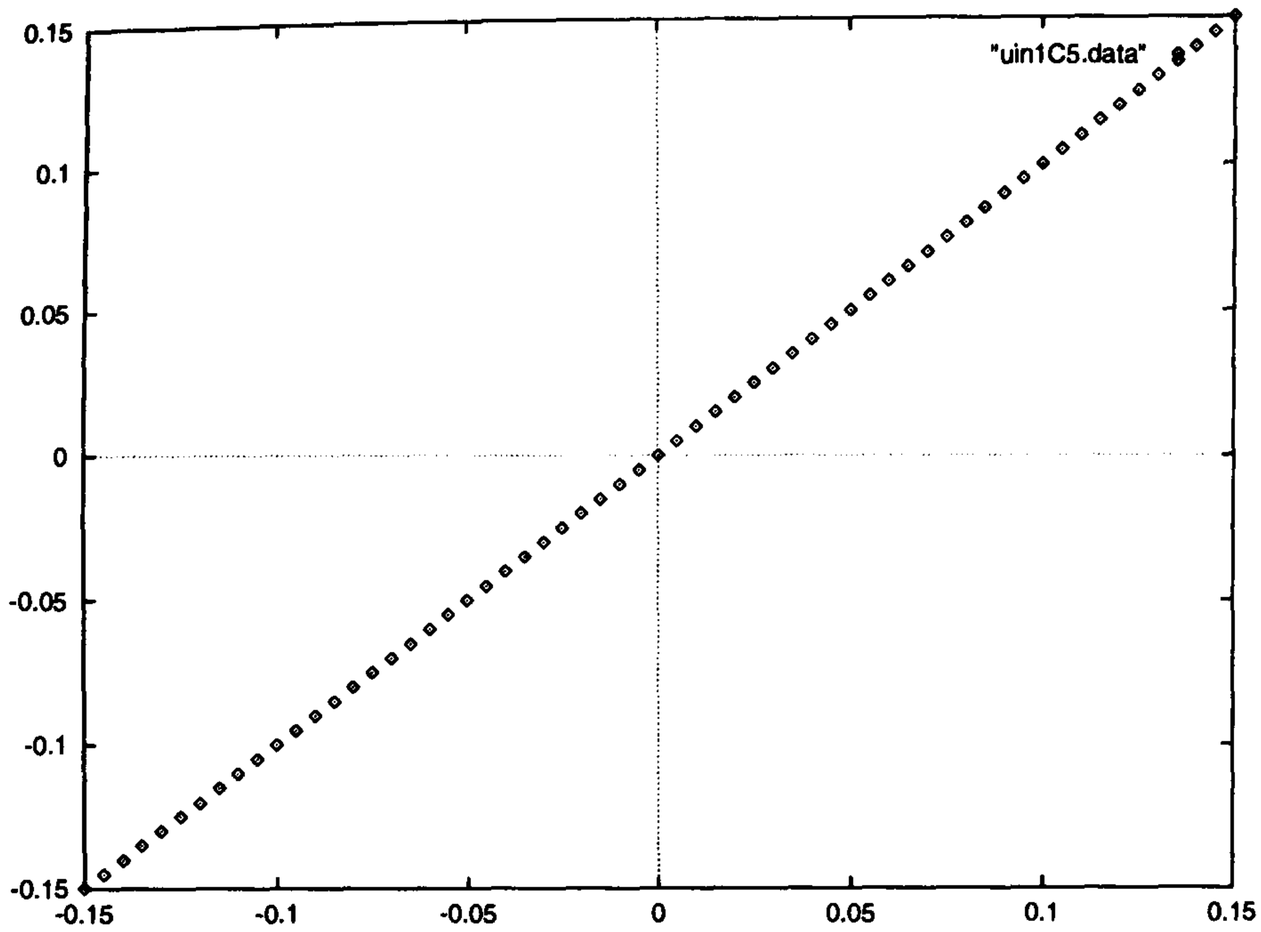
Spice-File:

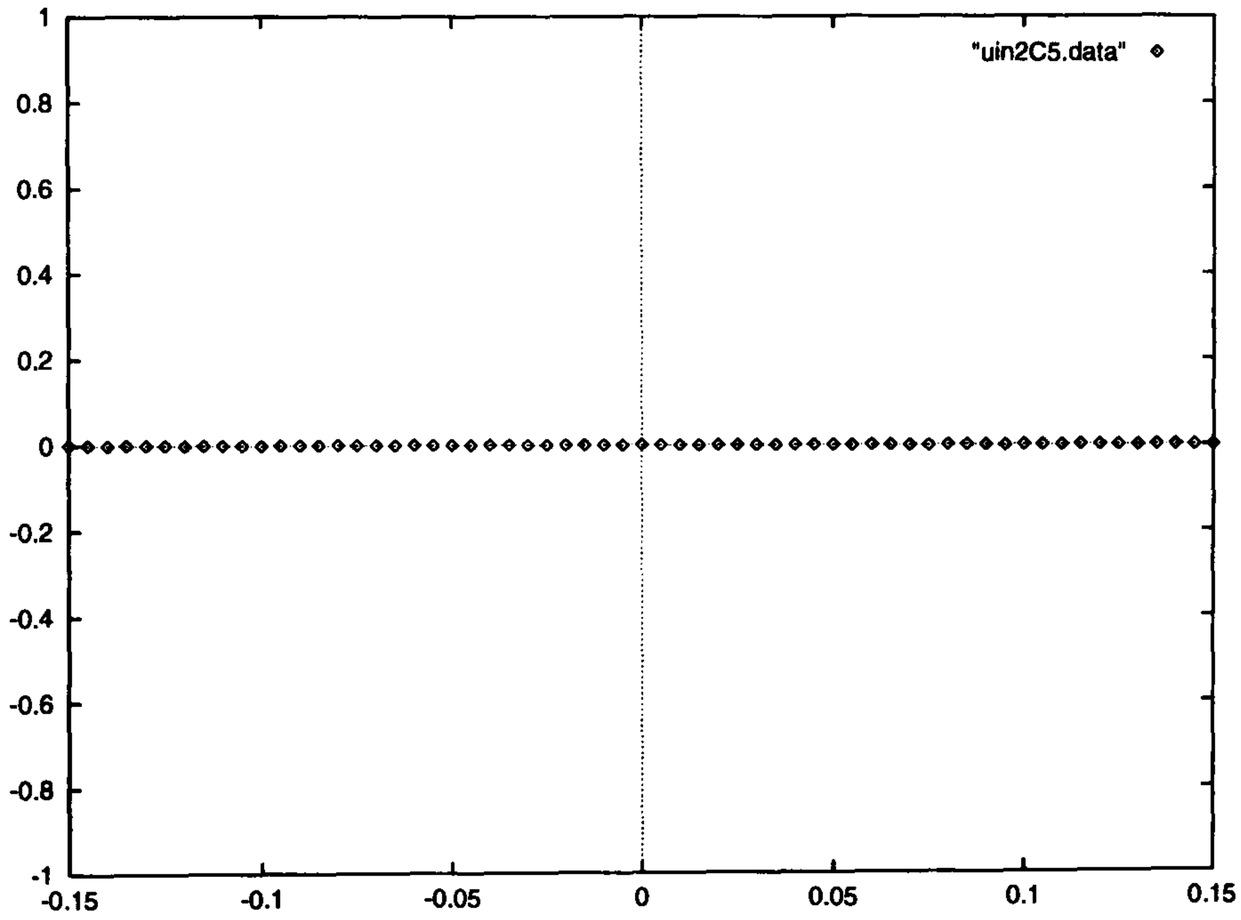
```
Operational Amplifier
vcc 5 0 10
vee 7 0 -10
vin 1 0
q1 2 1 4 tr1
q2 3 0 4 tr1
q3 4 6 7 tr1
q4 6 6 7 tr1
q5 5 3 8 tr1
q6 9 6 7 tr1
.model tr1 npn is=10f vaf=100
rc1 5 2 10k
rc2 5 3 10k
rr 5 6 20.4k
rs 8 9 4.3k
.dc vin -150m 150m 5m
.plot dc v(1,0)
.plot dc v(9,0)
.end
```

Black-Box Description:

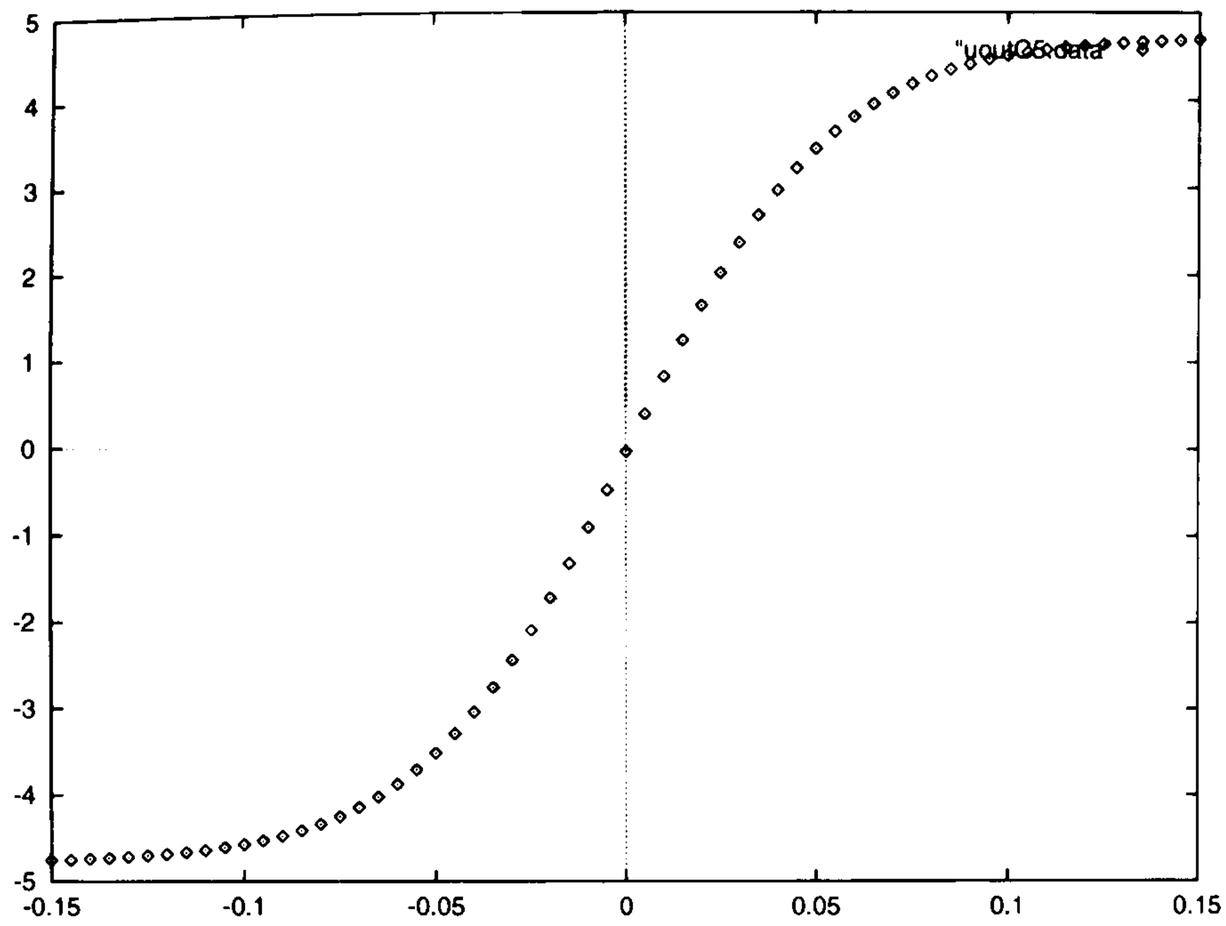


Input Data:


"uin1C5.data"
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.15
-0.1
-0.05
0
0.05
0.1
0.15


"uin2C5.data"
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.15
-0.1
-0.05
0
0.05
0.1
0.15

Output Data:

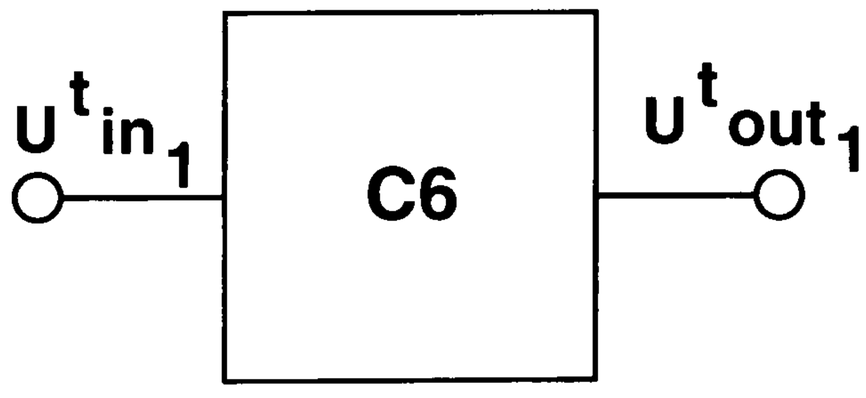


C.6 Circuit C6

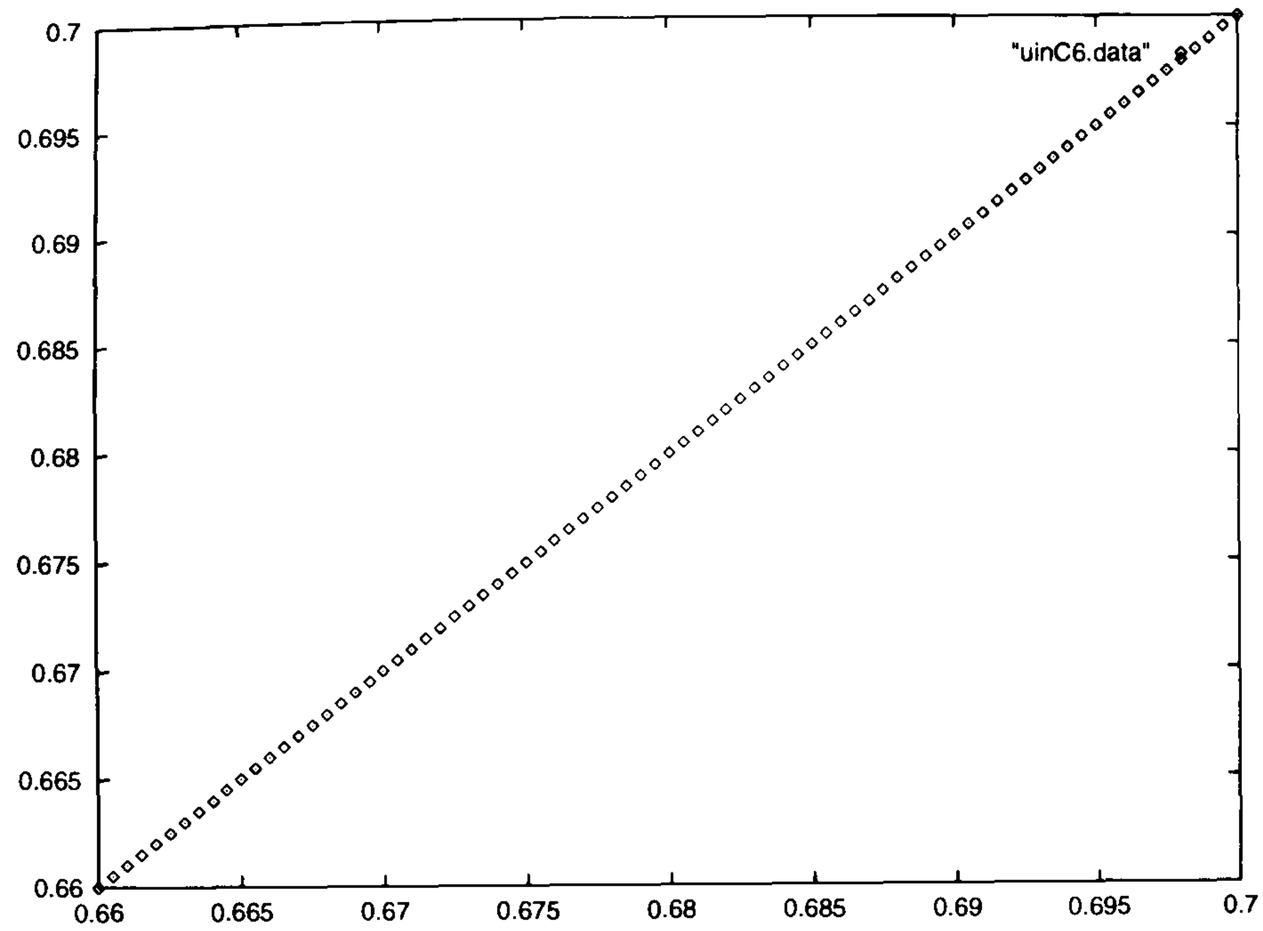
Spice-File:

```
Emitter Amplifier

vcc 4 0 5

vin 1 0 0.68

rref 3 0 1.62k

q1 2 1 0 tr1

q2 2 3 4 tr2

q3 3 3 4 tr2

.model tr1 npn is=10f vaf=100

.model tr2 pnp is=10f vaf=100

.dc vin 660m 700m 0.5m

.plot dc v(1,0)

.plot dc v(2,0)

.end
```

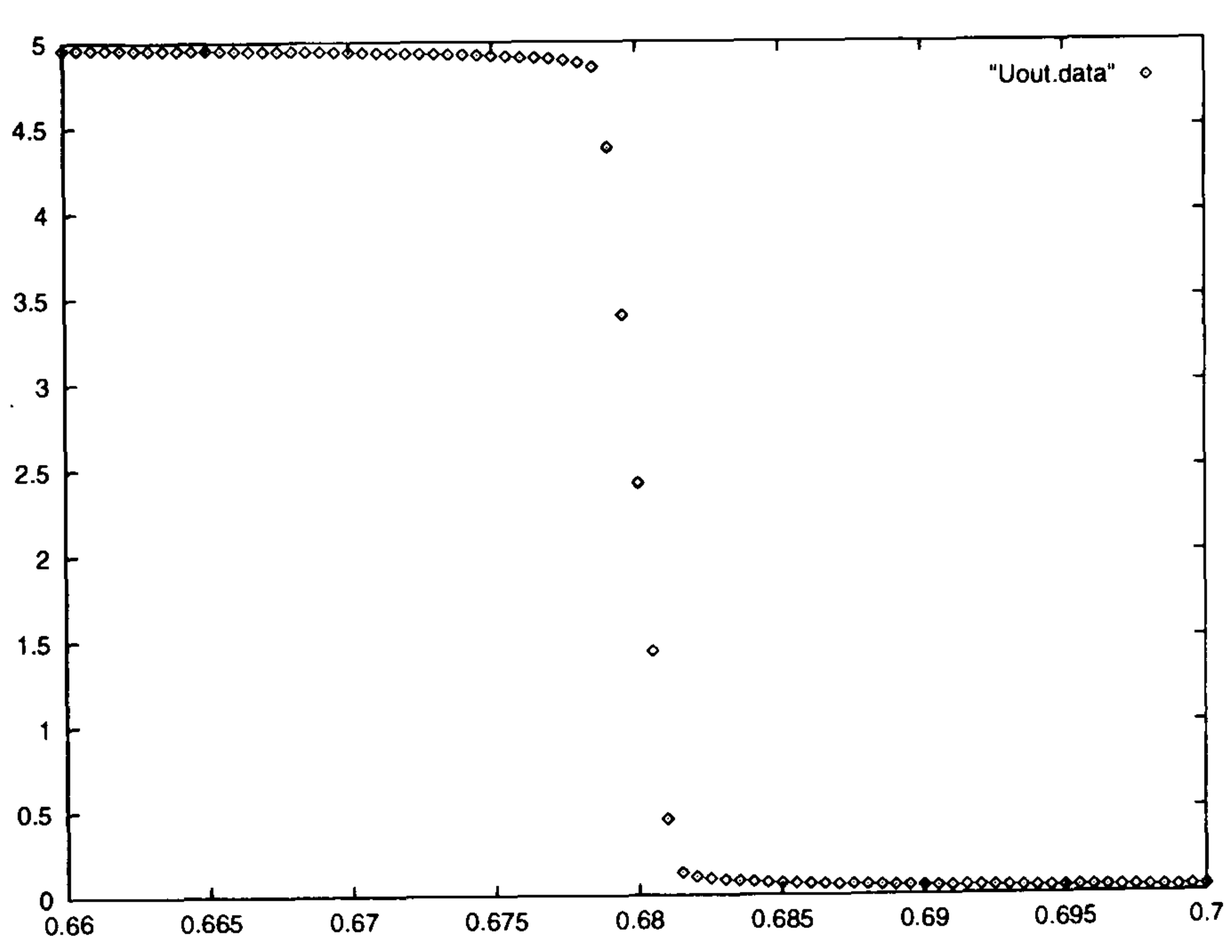
Black-Box Description:



Input Data:



Output Data:

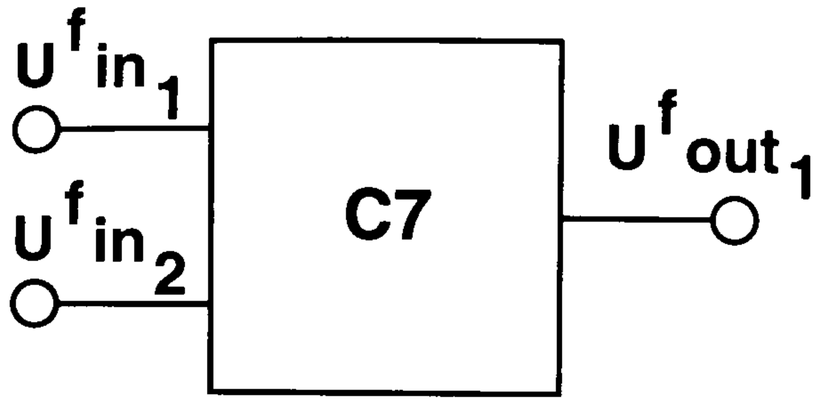


C.7 Circuit C7

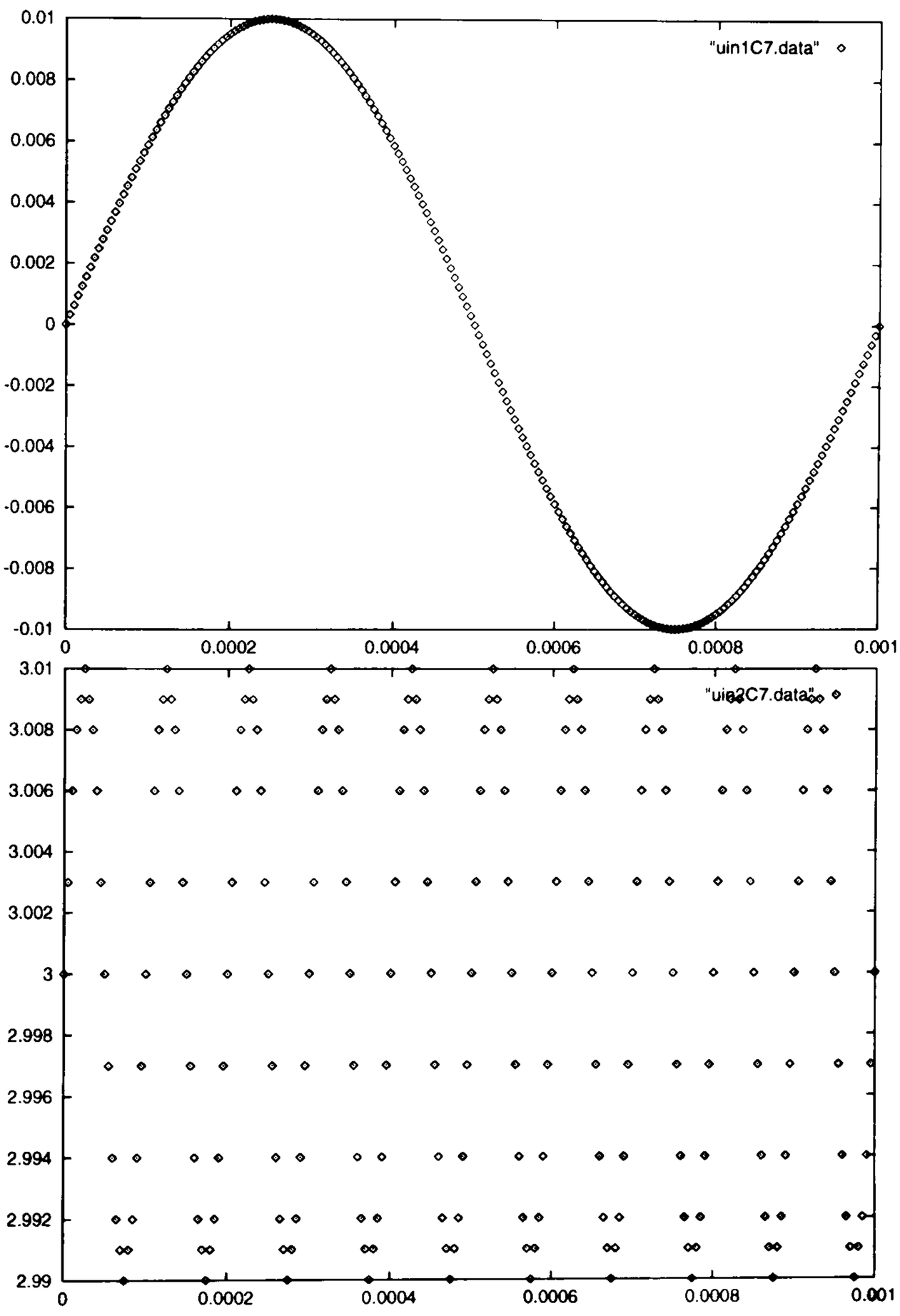
Spice-File:

```
Analog Multiplier
q1 6 8 9 tr1
q2 7 0 9 tr1
q3 2 4 6 tr1
q4 3 5 6 tr1
q5 2 5 7 tr1
q6 3 4 7 tr1
.model tr1 npn is=10f vaf=100
rc1 1 2 10k
rc2 1 3 10k
vcc 1 0 15
vee 10 0 -15
v1 8 0 sin 0 10m 1k
v2 4 0 sin 3 10m 10k
v3 5 0 3
iee 9 10 1m
.tran 5u 1m 0 5u
.plot tran v(8,0)
.plot tran v(4,0)
.plot tran v(2,3)
.end
```

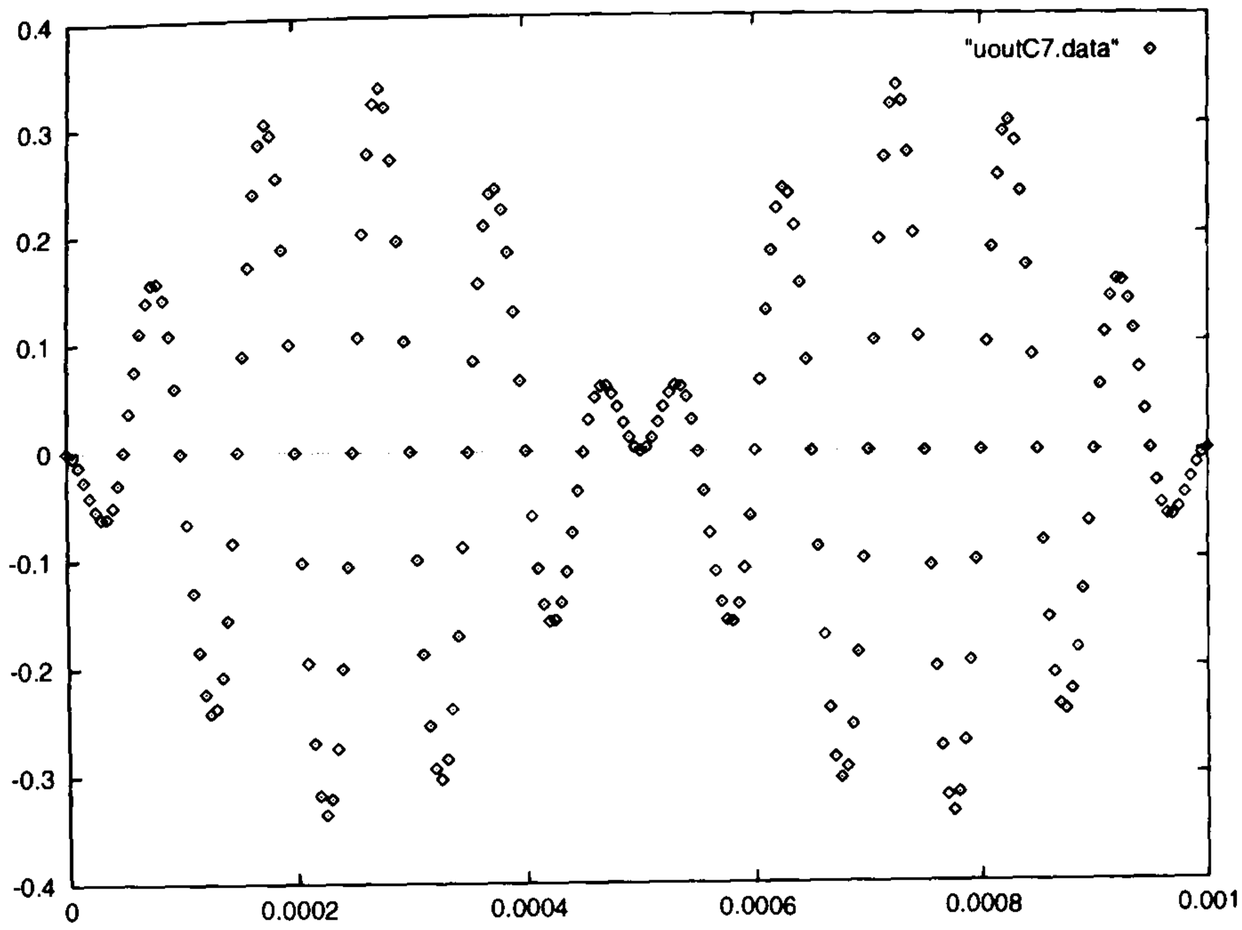
Black-Box Description:



Input Data:



Output Data:

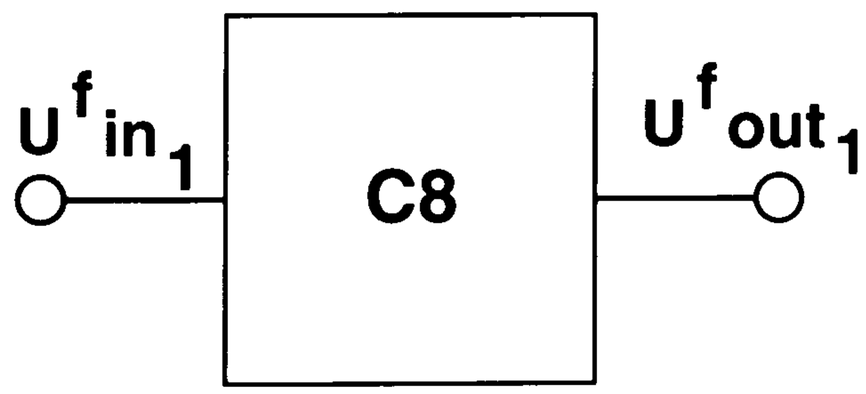

"uoutC7.data"
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.0002
0.0004
0.0006
0.0008
0.001

C.8 Circuit C8

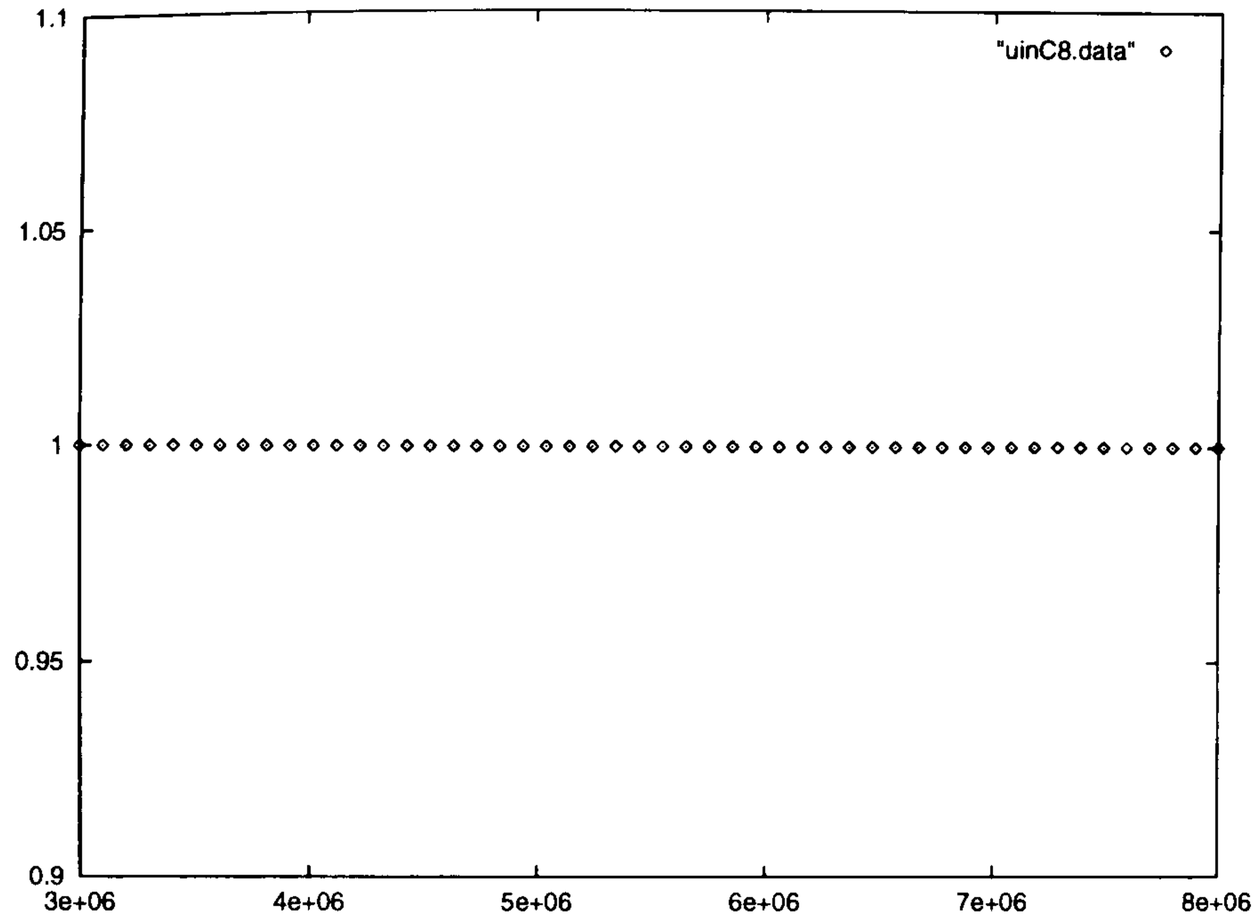
Spice-File:

```
Transconductance-C Filter

vinbs 1 0 AC 1

vinbp 5 0

r1 1 2 1g

r3 2 3 1g

r4 4 0 1g

c0 1 4 30p

c1 3 5 3p

c2 2 3 27p

e1 4 0 1*v(4,0) 2 0

g1 1 2 0.9425*v(1,2) 0 3

g2 3 2 0.9425*v(3,2) 0 4

.ac lin 50 3meg 8meg

.plot ac v(1,0)

.plot ac v(2,0)

.end
```

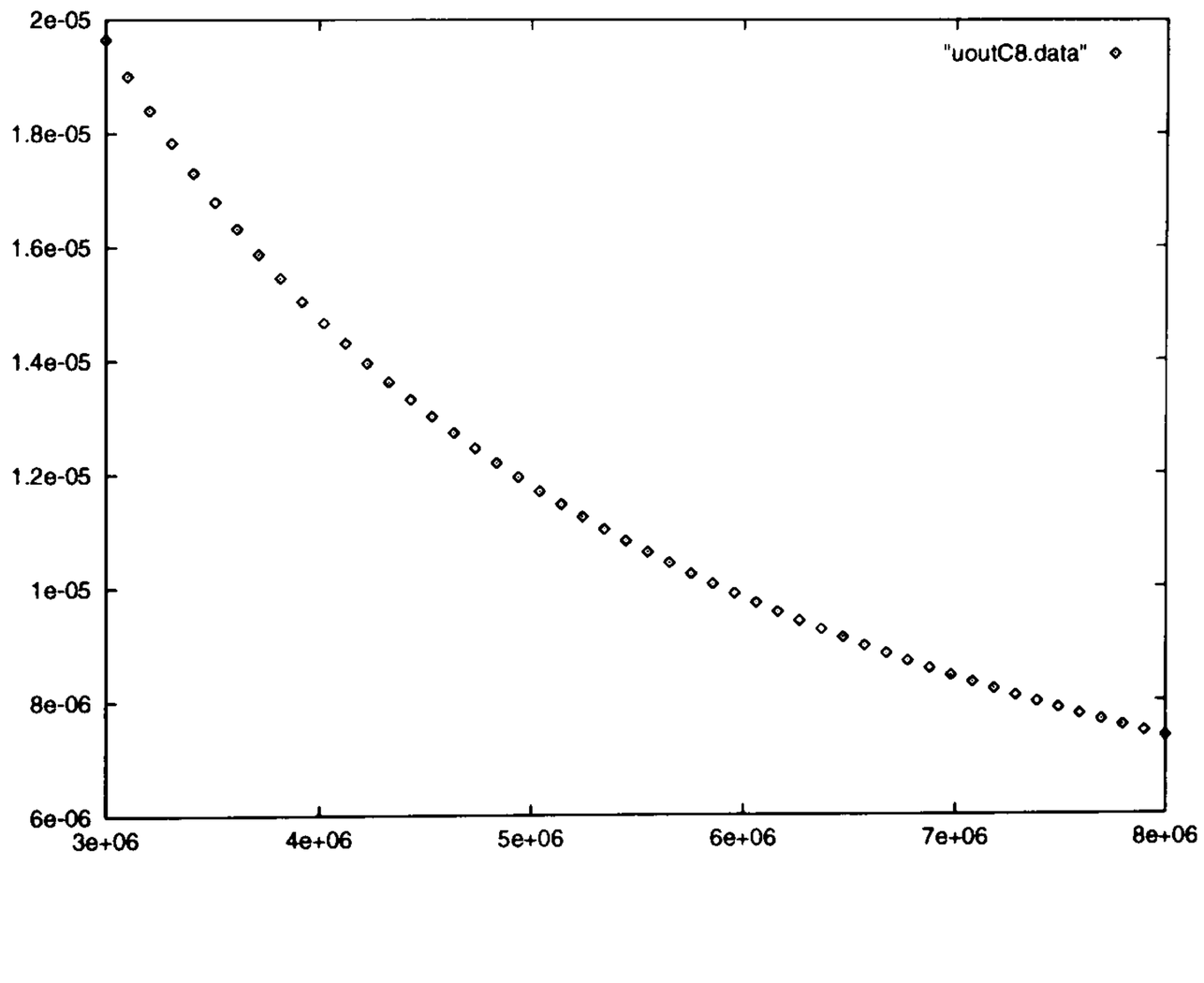
Black-Box Description:



Input Data:



Output Data:



Appendix D

Software of the CD

Most of the figures and all figures related to Fuzzy Relation Memories of the thesis are generated by programs. The computational expensive programs are written in C and $C++$. The user interface prototype software is written in Java (JDK 1.1: Java Development Kit 1.1 [JavaSoft, Sun Microsystems Inc., 1997]). The CD contains the source code and binaries (compiled for Intel Linux operating system).

D.1 Description of CD Contents

README gives an overview of of the CD content

Doc contains the dissertation in latex format. The directory is subdivided into 2 parts:

Part I contains the theoretical foundation documentation of the dissertation.

Part II contains the application documentation of the dissertation.

Soft contains the source code of the programs used in this dissertation.

Gnuplot3.5 GNUPLOT is a command-driven interactive function plotting program developed by Thomas Williams, Colin Kelley.

bin executable binaries for Intel Linux operating systems. The directory is subdivided into Chap5, Chap6, Chap7, Chap10, and AppA that contains software related to the chapters of the thesis.

jdk-1.1.3 Java Development Kit 1.1 of Sun

src contains source code for programs used in Chapter 5, Chapter 6, Chapter 7, Chapter 10, and Appendix A. With shell scripts the programs are configured.

Spice contains analogue circuit description in SPICE-format [Nagel, 1975].

D.2 Overview of CD Contents

```
.
|-- Doc
|    |-- Appendix
|    |    |-- AppendixDatabase
|    |    `-- AppendixFuzzy
|    |-- FrontPages
|    |-- PartI
|    |    |-- ConstraintRepresentation
|    |    |-- DifferentialEquation
|    |    |-- History
|    |    |-- Modelling
|    |    |-- SignalRepresentation
|    |    |-- SimulationExample
|    |    |    |-- diffequationRL
|    |    |    |    `-- simResults
|    |    |    |         `-- Sim1
|    |    |    |-- diffequationRLC
```

```
|   |   |   |   '-- simResults
|   |   |   |       |-- Sim1
|   |   |   |       |-- Sim2
|   |   |   |       '-- Sim3
|   |   |   '-- softqual
|   |   '-- ValueRepresentation
|   |-- PartII
|   |   |-- CellExtraction
|   |   |-- DatabaseExamples
|   |   |   '-- AmplifierExample
|   |   |       |-- Fuzzy
|   |   |       '-- Qual
|   |   |-- Framework
|   |   |-- PaperPrototype
|   |   |   |-- soft
|   |   |   '-- softqual
|   |   |-- Previous
|   |   '-- Spec
|   |       '-- InputSignal
|   '-- bin
|-- Soft
|   |-- Gnuplot3.5
|   |   '-- man
|   |-- bin
|   |   |-- AppA
|   |   '-- TFIArithmetic
|   |   |-- Chap10
|   |   |   '-- gofsas
|   |   |-- Chap4
```

```
|   |   |   |-- FuzzifyingFunction
|   |   |   |-- FuzzyRelation
|   |   |   |-- FuzzyRelationMemory
|   |   |   |-- QualitativeFRM
|   |   |   |-- TemporalFuzzifyingFunction.C
|   |   |   `-- TemporalFuzzifyingFunction.CPP
|   |   |-- Chap5
|   |   |   |-- FRMarithmetic
|   |   |   `-- Kettle
|   |   `-- Chap6
|   |       |-- InteractiveApproach
|   |       |-- InteractiveEvolutionaryAlgorithm
|   |       `-- NonInteractiveApproach
|   |-- jdk-1.1.3
|   |   |-- bin
|   |   |   |-- i386 -> i586
|   |   |   |-- i486 -> i586
|   |   |   |-- i586
|   |   |   |   `-- green_threads
|   |   |   `-- i686 -> i586
|   |   |-- i386 -> i586
|   |   |-- i486 -> i586
|   |   |-- i586
|   |   |   `-- green_threads
|   |   |-- i686 -> i586
|   |   |-- include
|   |   |   |-- genunix
|   |   |   `-- green_threads
|   |   |       `-- include
```

```
|   |   |-- lib -> .
|   |   `-- security
|   `-- src
|       |-- AppA
|       |   `-- TFIArithmetic
|       |       `-- data
|       |-- Chap10
|       |   `-- gofsas
|       |-- Chap12
|       |   `-- AmplifierExample
|       |       |-- Fuzzy
|       |       `-- Qual
|       |-- Chap4
|       |   |-- FuzzifyingFunction
|       |   |   `-- data
|       |   |-- FuzzyRelation
|       |   |   `-- data
|       |   |-- FuzzyRelationMemory
|       |   |   `-- data
|       |   |-- QualitativeFRM
|       |   |   `-- data
|       |   |-- TemporalFuzzifyingFunction.C
|       |   |   `-- data
|       |   `-- TemporalFuzzifyingFunction.CPP
|       |       `-- data
|       |-- Chap5
|       |   |-- FRMDerivation
|       |   |   `-- data
|       |   |-- FRMarithmetic
```

```
|       |   |   '-- data
|       |   '-- Kettle
|       |       '-- data
|       |-- Chap6
|       |   |-- InteractiveApproach
|       |   |   '-- data
|       |   |-- InteractiveEvolutionaryAlgorithm
|       |   |   '-- Samples
|       |   |       |-- diffequationRLC
|       |   |       |   |-- Sim1
|       |   |       |   |-- Sim2
|       |   |       |   |-- Sim3
|       |   |       |   '-- Sim4
|       |   |       |-- simResultsA
|       |   |       |   |-- ExactSim
|       |   |       |   |-- Sim1
|       |   |       |   |-- Sim2
|       |   |       |   |-- Sim3
|       |   |       |   |-- Sim4
|       |   |       |   |-- Sim5
|       |   |       |   |-- Sim6
|       |   |       |   '-- compare
|       |   |       |-- simResultsB
|       |   |       |   |-- ExactSim
|       |   |       |   |-- Sim1
|       |   |       |   |-- Sim2
|       |   |       |   |-- Sim3
|       |   |       |   |-- Sim4
|       |   |       |   |-- Sim5
```

```
|       |   |       |   |-- Sim6
|       |   |       |   `-- compare
|       |   |       |-- simResultsC
|       |   |       |   |-- Exact
|       |   |       |   `-- GA
|       |   |       `-- simResultsD
|       |   |-- NonInteractiveApproach
|       |   |   `-- data
|       |   |-- Numeric
|       |   |   |-- 1.OrdnungDiff
|       |   |   |   |-- Euler1
|       |   |   |   |-- Euler2
|       |   |   |   `-- RungeKutta
|       |   |   |-- 2.OrdnungDiff
|       |   |   |   |-- Euler1
|       |   |   |   |-- Euler2
|       |   |   |   `-- RungeKutta
|       |   |   |-- GA.Interval.nOrdnungDiff
|       |   |   |   `-- simResults
|       |   |   |       |-- Sim1
|       |   |   |       |-- Sim2
|       |   |   |       |-- Sim3a
|       |   |   |       |-- Sim3b
|       |   |   |       |-- Sim4
|       |   |   |       |-- Sim5
|       |   |   |       |-- Sim6
|       |   |   |       |   `-- compare
|       |   |   |       `-- Sim7
|       |   |   |           `-- compare
```

```
|       |   |   |-- Interval.n.OrdnungDiff
|       |   |   |-- NonInteractive
|       |   |   `-- n.OrdnungDiff
|       |   |       |-- RungeKutta
|       |   |       `-- RungeKuttaInterval
|       |   `-- SimpleInteractiveEvolutionaryAlgorithm
|       `-- Tools
|           |-- Comperator
|           |-- Differential
|           |   `-- funcXXX
|           |-- Integral
|           |-- TransformSignals
|           `-- TwoPoint2Function
`-- Spice
    |-- Cir1
    |   `-- Data
    |-- Cir10
    |   `-- Data
    |-- Cir2
    |   `-- Data
    |-- Cir3
    |   `-- Data
    |-- Cir4
    |   `-- Data
    |-- Cir5
    |   `-- Data
    |-- Cir6
    |   `-- Data
    |-- Cir7
```

```
|    '-- Data
|-- Cir8
|    '-- Data
|-- Cir9
|    '-- Data
|-- VCO
'-- VSum
```

Bibliography

[Bachmann et al., 1993] Bachmann, B., Bernardi, A., Klauck, C., and Schmidt, G. (1993). Design & KI. Deutsches Forschungszentrum für Küngstliche Intelligenz GmbH.

[Banzhaf et al., 1998] Banzhaf, W., Nordin, P., Keller, R., and Francone, F. (1998). *Genetic Programming — An Introduction.* Morgan Kaufmann Publishers, Inc., San Francisco, California.

[Beenker et al., 1993] Beenker, G., Conway, J., Schrooten, G., and Slenter, A. (1993). Analog CAD for Consumer ICs. In Huijsing, J., Van der Plassche, R., and Willy, S., editors, *Analog Circuit Design*, Boston. Kluwer Academic Publishers.

[Berleant and Kuipers, 1992] Berleant, D. and Kuipers, B. (1992). Combined qualitative and numerical simulation with Q3. In Faltings, B. and Struss, P., editors, *Recent Advances in Qualitative Physics.* MIT Press, Cambrideg, Massachusetts.

[Berleant and Kuipers, 1998] Berleant, D. and Kuipers, B. (1998). Qualitative and quantitative simulation: Bridging the gap. *Journal of Artificial Intelligence*, 95:215–255.

[Bonarini and Bontempi, 1994] Bonarini, A. and Bontempi, G. (1994). A qualitative simulation approach for fuzzy dynamical models. *ACM Transactions on Modelling and Computer Simulation*, 4(4):285–313.

[Bronstein and Semendjajew, 1991] Bronstein, I. and Semendjajew, K. (1991). *Taschenbuch der Mathematik.* B.G. Teubner Verlagsgesellschaft, Stuttgart. 25th edition.

[Buckley and Siler, 1988] Buckley, J. and Siler, W. (1988). *Fuzzy Numbers For Expert Systems*, pages 153–172. Volume Fuzzy Logic in Knowledge-Based Systems, Decision and Control of [Gupta and Yamakawa, 1988].

[Camposano, 1989] Camposano, R. (1989). Synthesing circuits from behavioral descriptions. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 8(2).

[Camposano et al., 1991] Camposano, R., Saunders, L., and Tabet, R. (1991). VHDL as input for high-level synthesis. *IEEE Design and Test of Computers*, 8(1):43–49.

[Charniak and McDermott, 1985] Charniak, E. and McDermott, D. (1985). *Introduction to Artificial Intelligence.* Addison-Wesley.

[Chen and Yang, 1995] Chen, J. and Yang, A. (1995). Style: A statistical design approach based on nonparametric performance macromodelling. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):794–802.

[Cubert and Fishwick, 1997] Cubert, R. M. and Fishwick, P. A. (1997). Moose: An object-oriented multimodelling and simulation application framework. *Simulation.*

[Davis, 1989] Davis, E. (1989). Order of magnitude reasoning in qualitative differential equations. In Weld, D. S. and de Kleer, J., editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 422–434. Technical Report, New York University Computer Science Department.

[de Kleer, 1977] de Kleer, J. (1977). Multiple representations of knowledge in a mechanics problem solver. In *Proceedings of the International Jornal Conference*

in Artificial Intelligence 1977, pages 299–304, Cambrideg, Massachusetts. MIT Press.

[de Kleer, 1984] de Kleer, J. (1984). How circuits work. *Journal of Artificial Intelligence*, 24:205–280.

[de Kleer and Brown, 1984] de Kleer, J. and Brown, J. S. (1984). Qualitative physics based on confluences. *Journal of Artificial Intelligence*, 24:7–83. Also in *Readings in Knowledge Representation*, Brachman and Levesque, editors, Morgan Kaufmann, 1985, pages 88-126.

[de Kleer and Brown, 1986] de Kleer, J. and Brown, J. S. (1986). Theories of causal ordering. *Journal of Artificial Intelligence*, 29:33–61.

[DeCoste, 1991] DeCoste, D. M. (1991). Dynamic across-time measurement interpretation. *Journal of Artificial Intelligence*, 51(1-3):273–341.

[Degrauwe et al., 1989] Degrauwe, M., Goffart, B., Meixenberger, C., Pierre, M., Litsios, J., Rijmenants, J., Nys, O., Dijkstra, E., Joss, B., Meyvaert, M., Schwarz, T., and Pardoen, M. (1989). Towards an analog system design environment. *IEEE Journal of Solid-Sate Circuits*, 24(3).

[Degrauwe et al., 1987] Degrauwe, M., Nys, O., Dijkstra, E., Rijmenants, J., Bitz, S., Goffart, B., Vittoz, E., Cserveny, S., Meixenberger, C., VanDerStappen, G., and Oguey, H. (1987). IDAC: An interactive design tool for analog CMOS circuits. *IEEE Journal of Solid-Sate Circuits*, SC-22(6).

[Dong and Shah, 1987] Dong, W. and Shah, H. (1987). Vertex method for computing functions of fuzzy variables. *Fuzzy Sets and Systems*, 24:65–78.

[Dong et al., 1985] Dong, W., Shah, H., and Wong. F. (1985). Fuzzy computations in risk and decision analysis. *Civil Engineering Systems*, 2:201–208.

[Dubois and Prade, 1980] Dubois, D. and Prade, H. (1980). *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, Inc., San Diego.

[Duchene et al., 1993] Duchene, P., Declercq, M., Goffart, B., and Novak, M. (1993). Analog circuits implementation on CMOS semi-custom arrays. *IEEE Journal of Solid-Sate Circuits*, 28(7):872–874.

[Duprè, 1987] Duprè, J. (1987). *The latest on the best. Essays on Evolution an Optimality.* MIT Press, Cambridge, MA.

[El-Turky and Perry, 1989] El-Turky, F. and Perry, E. (1989). BLADES: An artificial intelligence approach to analog circuit design. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 8(6).

[Fares and Bozena, 1995] Fares, M. and Bozena, K. (1995). FPAD: A fuzzy nonlinear programming approach to analog circuit design. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):785–793.

[Farquhar, 1993] Farquhar, A. (1993). *Automated Modelling of Physical Systems in the Presence of Incomplete Knowledge.* PhD thesis, University of Texas at Austin.

[Fishwick, 1991] Fishwick, P. A. (1991). Fuzzy simulation: Specifying and identifying qualitative models. *International Journal of General System*, 19:295–316.

[Fishwick, 1997] Fishwick, P. A. (1997). A visual object-oriented multimodelling design approach for physical modelling. *ACM Transactions on Modelling and Computer Simulation.*

[Forbus, 1984] Forbus, K. (1984). Qualitative process theory. *Journal of Artificial Intelligence*, 24:85–168.

[Forbus, 1993] Forbus, K. (1993). Qualitative process theory: twelve years after. *Journal of Artificial Intelligence*, 59:115–123.

[Forbus, 1987] Forbus, K. D. (1987). Interpreting observations of physical systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:350–359.

[Fox, 1993] Fox, J. (1993). A higher level of synthesis. *IEEE Spectrum*, pages 43–47.

[Fujita et al., 1986] Fujita, T., Mori, H., and Mitsumoto, K. (1986). Artificial intelligent approach to VLSI design. In Goto, S., editor, *Design Methodologies*, pages 441–464. North-Holland Publishing Company.

[Geyer-Schulz, 1995] Geyer-Schulz, A. (1995). *Fuzzy Rule-Based Expert Systems and Genetic Machine Learning.* Physica-Verlag, Heidelberg, 2nd edition.

[Gielen et al., 1990] Gielen, G., Walscharts, H., and Sansen, W. (1990). Analog circuit design optimization based on symbolic simulation and simulated annealing. *IEEE Journal of Solid-Sate Circuits*, 25(3):707–713.

[Givens and Tahani, 1987] Givens, J. and Tahani, H. (1987). An improved method of performing fuzzy arithmetic for computer vision. In *Proceedings of North American Information Processing Society (NAFIPS)*, pages 275–280, Purdue University, West Lafayette, IN.

[Grimbleby, 1990] Grimbleby, J. (1990). *Computer-aided Analysis and Design of Electronic Networks.* Pitman Publishing, London.

[Grimm et al., 1995] Grimm, C., Öhler, P., and Waldschmidt, K. (1995). Modellierung gemischt analog/digitaler Systeme zur Entwurfsunterstützung. In *Hardwarebeschreibungssprachen und Modellierungsparadigmen*, pages 1.1–1.2. ITG Fachgruppe 5.2.2, GI Fachausschuš 2.5.

[Gupta et al., 1988] Gupta, M., Knopf, G., and Nikiforuk, P. (1988). *Sinusoidal-Based Cognitive Mapping Functions.* North-Holland Publishing Company. Amsterdam.

[Gupta et al., 1979] Gupta, M., Ragade, R., and Yager, R., editors (1979). *Advances in Fuzzy Set Theory and Applications.* North Holland Publishing Company.

[Gupta and Yamakawa, 1988] Gupta, M. and Yamakawa, T., editors (1988). *Fuzzy Logic in Knowledge-Based Systems, Decision and Control.* North-Holland.

[Gyooseok and Fishwick, 1997] Gyooseok, K. and Fishwick, P. A. (1997). A method for resolving the consistency problem between rule-based and quantitative models using fuzzy simulation. *IEEE Transactions on Systems, Man and Cybernetics.*

[Harjani et al., 1989] Harjani, R., Rutenbar, R., and Carley, L. (1989). OASYS: A framework for analog circuit synthesis. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 8(12).

[Harvey et al., 1992] Harvey, J., Elmasry, M., and Leung, B. (1992). STAIC: An interactive framework for synthesizing CMOS and BICMOS analog circuits. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 11(11).

[Horrocks and Arslan, 1995] Horrocks, D. and Arslan, T. (1995). The design of analogue and digital filters using genetic algorithms. *Proceedings of 15th Saraga Collogium on Digital and Analogue Filters and Filtering Systems*, pages 7.1–7.5.

[Horrocks and Khalifa, 1994] Horrocks, D. and Khalifa, Y. (1994). Genetically derived filters using preferred values components. *Proceedings of IEE Collogium on Linear Analogue Circuits and Systems.*

[Horrocks and Khalifa, 1995] Horrocks, D. and Khalifa, Y. (1995). Genetically evolved FDNR and Leap-Frog filters using preferred component values. *Proceedings of European Conference on Circuit Theory and Design*, pages 359–362.

[Horrocks and Spittle, 1993] Horrocks, D. and Spittle, M. (1993). Component value selection for active filters using genetic algorithms. *Proceedings of IEE Workshop on Natural Algorithms in Signal Processing*, 1:13.1–13.6.

[Huang and Fan, 1993] Huang, Y. and Fan, L. (1993). A fuzzy logic-based approach to building efficient fuzzy rule-based expert systems. *Computer Chemical Engineering*, 17(2):188–192.

[Iwasaki and Simon, 1986] Iwasaki, Y. and Simon, H. (1986). Theories of causal ordering: Reply to de Kleer and Brown. *Journal of Artificial Intelligence*, 29:63–72.

[Jamshidi et al., 1993] Jamshidi, M., Vadiee, N., and Ross, T., editors (1993). *Fuzzy rule-based expert systems — Part I and II*, volume Fuzzy Logic and Control: Software and Hardware Applications, chapter 4 and 5. Prentice Hall, Englewood Cliffs, New Jork.

[JavaSoft, Sun Microsystems Inc., 1997] JavaSoft, Sun Microsystems Inc. (1997). Java dokumentation. http://java.sun.com/docs/index.html.

[Karr and Gentry, 1993] Karr, C. and Gentry, E. (1993). Fuzzy control of pH using generic algorithms. *IEEE Transactions of Fuzzy Systems*, 1(1):46–53.

[Kaufmann, 1975] Kaufmann, A. (1975). *Introduction to the Theory of Fuzzy Supsets.* Academic Press, New York, vol. 1 edition.

[Kaufmann and Gupta, 1991] Kaufmann, A. and Gupta, M. (1991). *Introduction to Fuzzy Arithmetic – Theory and Applications.* Van Nostrand Reinhold, New York.

[Kerre and van Schooten, 1988] Kerre, E. E. and van Schooten, A. (1988). *A Deeper Look on Fuzzy Numbers From a Theoretical as Well as From a Practical Point of View*, pages 173–196. Volume Fuzzy Logic in Knowledge-Based Systems, Decision and Control of [Gupta and Yamakawa, 1988].

[Kleiber and Kulpa, 1995] Kleiber, M. and Kulpa, Z. (1995). Computer-assisted hybrid reasoning in simulation and analysis of physical systems. *Computer Assisted Mechanics and Engineering Science*, 2:165–186.

[Koh et al., 1990] Koh, H., Sequin, C., and Gray, P. (1990). OPASYN: A compiler for CMOS operational ampifiers. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 9(2).

[Kosko, 1992] Kosko, B. (1992). Fuzzy systems as universal approximators. *IEEE International Conference on Fuzzy Systems*, pages 1153–1162.

[Kosko, 1994] Kosko, B. (1994). Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, 43(11):1329–1333.

[Koza et al., 1996] Koza, J. R., Bennett III, F. H., Andre, D., and Keane, M. A. (1996). Automated design of both the topology and sizeing of analog electrical circuits using genetic programming. In Gero, J. S. and Sudweeks, F., editors, *Artificial Intelligence in Design 1996*, pages 151–170. Kluwer Academic Publishers, Boston, USA.

[Kuipers, 1993a] Kuipers, B. (1993a). Qualitative simulation: then and now. *Journal of Artificial Intelligence*, 59:133–140.

[Kuipers, 1993b] Kuipers, B. (1993b). Reasoning with qualitative models. *Journal of Artificial Intelligence*, 59:125–132.

[Kuipers, 1984] Kuipers, B. J. (1984). Commmonsense reasoning about causality: Deriving behavior from structure. *Journal of Artificial Intelligence*, 24:169–204.

[Kuipers, 1986] Kuipers, B. J. (1986). Causal simulation. *Journal of Artificial Intelligence*, 29:289–338.

[Kuipers, 1994] Kuipers, B. J. (1994). *Qualitative Reasoning - Modelling and Simulation with Incomplete Knowledge.* MIT Press, Cambrideg, Massachusetts.

[Kuipers and Åström, 1994] Kuipers, B. J. and Åström, K. (1994). The composition and validation of heterogeneous control laws. *Automatica*, 30(2):233–249.

[Kurup and Abbasi, 1995] Kurup, P. and Abbasi, T. (1995). *Logic synthesis using SYNOPSIS.* Kluwer Academic Publishers, Boston.

[Lai et al., 1988] Lai, J., Kueng, J., Chen, H., and Fermandez, F. (1988). ADOPT – a CAD system for analog circuit design. In *Proceedings IEEE Custom Integrated Circuit Conference (CICC)*, pages 3.2.1–3.2.4.

[Larkin and Simon, 1987] Larkin, J. and Simon, H. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11.

[Lindfield and Penny, 1989] Lindfield, G. and Penny, J. (1989). *Microcomputers in Numerical Analysis.* John Wiley & Sons, New York.

[Lunze, 1995] Lunze, J. (1995). *Künstliche Intelligenz für Ingenieure — Band 2: Technische Anwendungen.* Oldenbourg Verlag, München.

[Malavasi et al., 1993] Malavasi, E., Chang, H., Sangiovanni-Vincentelli, A., Charbon, E., Choudhury, U., Felt, E., Jusuf, G., Liu, E., and Neff, R. (1993). A top-down, constraint-driven design methodology for analog integrated circuits. In Huijsing, J., Van der Plassche, R., and Willy, S., editors, *Analog Circuit Design*, pages 285–322, Boston. Kluwer Academic Publishers.

[Martschew, 1988] Martschew, E. (1988). Processes: Qualitative process theory and beyond. In Früchtenicht, H., Güsgen, H., Hrycej, T., and Struss, P., editors, *Technische Expertensysteme: Wissensrepräsentation und Schlußfolgerungsverfahren*, pages 55–79. Oldenbourg Verlag, München, Germany.

[Maulik and Carley, 1991] Maulik, P. and Carley, L. (1991). Automating analog circuit design using constrained optimization techniques. In *IEEE, International Conference on Computer Aided Design (ICCAD)*, pages 390–393, Santa Clara, California.

[Maulik et al., 1992] Maulik, P., Carley, L., and Rutenbar, R. (1992). A mixed-integer nonlinear programming approach to analog circuit synthesis. In *29th ACM/IEEE Design Automation Conference*, pages 698–704.

[Mc Neill and Freiberger, 1994] Mc Neill, D. and Freiberger, P. (1994). *Fuzzy Logic — The Revolutionary Computer Technology That is Changing Our World.* Simon & Schuster, New York, 1st edition.

[Mehranfar, 1991] Mehranfar, S. (1991). A technology-independent appraich to custom analog cell generation. *IEEE Journal of Solid-Sate Circuits*, 26(3):386–393.

[Milne, 1991] Milne, R. (1991). Second generation expert systems: The application gap. In *Proceeding 11th International Conference on Expert Systems and their Application (General Conference On 2nd Generation Expert Systems)*, pages 259–264, Avignon, France.

[Mizumoto and Tanaka, 1979] Mizumoto, M. and Tanaka, K. (1979). Some properties of fuzzy numbers. In [Gupta et al., 1979], pages 153–164.

[Mueller-Glaser and Bortolazzi, 1990] Mueller-Glaser, K. and Bortolazzi, J. (1990). An approach to coputer-aided specification. *IEEE Journal of Solid-Sate Circuits*, 25(2):335–345.

[Nagel, 1973] Nagel, L. (1973). SPICE. Berkeley, University of California, Electronic Research Laboratory.

[Nagel, 1975] Nagel, L. (1975). SPICE2: A computer program to simulate semiconductor circuits. Berkeley, University of California, Electronic Research Laboratory.

[Negoita, 1985] Negoita, C. V. (1985). *Expert Systems and Fuzzy Systems.* Benjamin/Cummings Publishing Company, Inc., New York.

[Neville and Weld, 1994] Neville, D. and Weld, D. (1994). Innovative design as systematic search. In *Working Notes of the AAAI Fall Symposium on Design from Physical Principles*, pages 737–72.

[Nye et al., 1988] Nye, W., Riley, D., Sangiovanni-Vincentelli, A., and Tits, A. (1988). DELIGHT.SPICE: An optimization-based system for the design of integrated circuits. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 7(4).

[Onodera et al., 1990] Onodera, H., Kanbara, H., and Tamaru, K. (1990). Operational amplifier compilation with performance optimization. *IEEE Journal of Solid-Sate Circuits*, 25(2):466–473.

[Pisan, 1995a] Pisan, Y. (1995a). Visual reasoning with graphs. http://www.cs.nwu.edu/ yusuf/.

[Pisan, 1995b] Pisan, Y. (1995b). A visual routines base model of graph understanding. In *Processings of the 17th Annual Conference of the Cognitive Science Society.*

[Raiman, 1991] Raiman, O. (1991). Order of magnitude reasoning. *Journal of Artificial Intelligence*, 51(1-3):11–38.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionsstrategie.* Frommann-Holzboog-Verlag, Stuttgart.

[Rechenberg, 1994] Rechenberg, I. (1994). *Evolutionsstrategie '93.* Frommann-Holzboog-Verlag, Stuttgart.

[Reich, 1997a] Reich, C. (1997a). Fuzzy similarity. internally published at FH-Furtwangen.

[Reich, 1997b] Reich, C. (1997b). Simulation of analogue circuits using fuzzy curves. In Reuch, B., editor, *Computational Intelligence - Theory and Applications*, Berlin. International Conference, 5th Fuzzy Days, Proceedings, Springer.

[Reich, 1998] Reich, C. (1998). Fuzzy Relational Memories (FRMs) used to specify analogue circuits — modelling nonlinear systems not known exactly. In Selvaraj, H. and B., V., editors, *ICCIMA: International Conference on Computational Intelligence and Multimedia Applications*, Singapore. International Conference on Computational Intelligence and Multimedia Applications, World Scientific.

[Rosenman and Gero, 1994] Rosenman, M. and Gero, J. (1994). The what, the how, and the why in design. *Journal of Artificial Intelligence*, 8:199–218.

[Ross, 1995] Ross, T. (1995). *Fuzzy Logic With Engineering Applications.* McGraw-Hill, New York.

[Rutenbar, 1993] Rutenbar, R. (1993). Analog design automation: Where are we? Where are we going? In *Proceedings IEEE Custom Integrated Circuits Conference*, pages 13.1.1–13.1.7.

[Scheichenzuber, 1990] Scheichenzuber, J. (1990). Global hardware synthsis from behavioral dataflow descriptions. In *Proceedings of the 27th Design Automation Conference*, pages 456–461. IEEE.

[Schoeneburg et al., 1994] Schoeneburg, E., Heinzmann, F., and Feddersen, S. (1994). *Genetische Algorithmen und Evolutionsstrategien.* Addison-Wesley Publishing, Bonn.

[Schwefel, 1995] Schwefel, H.-P. (1995). *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology Series. John Wiley & Sons, New York.

[Sentovich et al., 1992] Sentovich, E., Singh, K., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P., Brayton, R., and Sangiovanni-Vincentelli, A.

(1992). SIS: A system for wequential circuit synthesis. *Technic Report UCB/ERL M92/41.*

[Sgouros, 1993] Sgouros, N. (1993). *Representing Physical and Design Knowledge in Innovative Design.* PhD thesis, Graduate School of Northwestern University, Evanston, Illinois.

[Shen and Leitch, 1993] Shen, Q. and Leitch, R. (1993). Fuzzy qualitative simulation. *IEEE Transactions onf Systems, Mans, and Dybernetics*, 23(4):1038–1061.

[Shyu and Sangiovanni-Vincentelli, 1988] Shyu, J. and Sangiovanni-Vincentelli, A. (1988). ECSTASY: A new environment for ic design optimization. In *Proceedings IEEE International Conference of Computer-Aided Design (ICCAD)*, pages 484–487.

[Smedley, 1996] Smedley, T. J. (1996). A high-level visual language for the graphical description of digital circuits. http://www.computer.org/conferen/vl95/ieee/smedley.ps.gz.

[Sommer and Henning, 1995] Sommer, R. and Henning, E. (1995). Application of computer algebra methods to analog circuit sizing. In *European Conference on Circuit Theory and Design*, Istanbul, Turkey.

[Sugeno, 1985] Sugeno, M., editor (1985). *Industrical Applications of Fuzzy Control.* North-Holland, Amsterdam.

[Sugeno and Yasukawa, 1993] Sugeno, M. and Yasukawa, T. (1993). A fuzzy logic-based approach to qualiative modelling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31.

[Swings and Sansen, 1993] Swings, K. and Sansen, W. (1993). ARIADNE: A constraint-based approach to computer-aided synthesis and modelling of analogue integrated circuits. In *Analog Integrated Circuits and Signal Processing 3*, pages 197-215, Boston.

[Szentirmai, 1977] Szentirmai, G. (1977). FILSYN – A general purose filter synthesis program. In *Proceedings of the IEEE*, pages 1443–1458.

[Tabachneck et al., 1994] Tabachneck, H. J., Leonardo, A. M., and Simon, H. A. (1994). How does an expert use a graph? A model of visual and verbal inferencing in economics. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, pages 842–847, Hillsdale, New York. Lawrence Erlbaum Associates.

[Takagi and Hayashi, 1991] Takagi, H. and Hayashi, I. (1991). Neural networks-driven fuzzy reasoning. *International Journal of Approximate Reasoning*, 5:191–212.

[Tong, 1979] Tong, R. (1979). The construction and evaluation of fuzzy models. In [Gupta et al., 1979], pages 559–576.

[Toumazou and Makris, 1995] Toumazou, C. and Makris, C. (1995). Analog IC design automation: Part i – automated circuit generation: New concepts and methods. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 14(2):218–238.

[VHDL-A Standard 99, 1999] VHDL-A Standard 99 (1999). *IEEE Standard VHDL-A Language Reference Manual. IEEE Std 1076.1-1999 (approved but not published)*. The Institute of Electrcal and Electronics Engineers, New York, USA.

[VHDL Standard 93, 1994] VHDL Standard 93 (1994). *IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-1993*. The Institute of Electrcal and Electronics Engineers, New York, USA.

[Wei, 1988] Wei, R.-S. (1988). BECOME: Behavior level circuit synthesis based on structure mapping. In *Proceedings of the 25th Design Automation Conference*, pages 409–414. IEEE.

[Werthner, 1994] Werthner, H. (1994). *Qualitative Reasoning — Modelling and Generation of Behavior.* Springer-Verlag, Wien.

[Wielinga and Guus, 1997] Wielinga, B. and Guus, S. (1997). Configuration-design problem solving. *IEEE Expert Intelligent Systems And Their Applications*, 12(2):49–56.

[Williams, 1984] Williams, B. (1984). Qualitative analysis of MOS circuits. *Journal of Artificial Intelligence*, 24:281–346.

[Williams, 1986] Williams, B. C. (1986). Doing time: Putting qualitative reasoning on firmer ground. In *Proceeding of AAAI86*, pages 105–113, Philadelphia, PA. AAAI.

[Williams, 1995] Williams, J., editor (1995). *The Art and Science of Analog Circuit Design.* SMTnet, Dallas, Texas, 2nd edition.

[Wittmann et al., 1994] Wittmann, R., Schardein, W., Hosticka, B., Schanz, M., Vahrmann, R., and Kern, S. (1994). Application-independent hierarchical synthesis methodology for analogue circuits. Association for Computing (ACM).

[Wood et al., 1992] Wood, K., Otto, K., and Antonsson, E. (1992). Engineering design calculations with fuzzy parameters. *Fuzzy Sets and Systems*, 52:1–20.

[Yager et al., 1987] Yager, R., Ovchinnikow, S., Tong, R., and Nguyen, H., editors (1987). *Fuzzy Sets And Applications: Selected Papers by Lotfi A. Zadeh.* John Wiley & Sons, Ney York.

[Yang et al., 1993] Yang, H., Yao, H., and Jones, J. (1993). Calculating functions of fuzzy numbers. *Fuzzy Sets and Systems*, 55:273–283.

[Yasunobu and Miyamoto, 1985] Yasunobu, S. and Miyamoto, S. (1985). Automatic train operation system by predictive fuzzy control. In [Sugeno, 1985], pages 1–18.

[Yip, 1991] Yip, K. M.-k. (1991). *KAM — A System for Intelligently Guiding Numerical Experimentation by Computer.* The MIT Press.

[Zadeh, 1965] Zadeh, L. A. (1965). *Fuzzy Sets*, pages 338–353. Volume Fuzzy Sets And Applications: Selected Papers by Lotfi A. Zadeh of [Yager et al., 1987]. pages in book: 29-44.

[Zadeh, 1971] Zadeh, L. A. (1971). Similarity relations and fuzzy orderings. *Information Science*, 3:177–200.

[Zadeh, 1972] Zadeh, L. A. (1972). A rationale for fuzzy control. *Journal of Dynamic System Measure Contral Transision ASME*, 94:3–4.

[Zadeh, 1973] Zadeh, L. A. (1973). *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes*, pages 28–44. Volume Fuzzy Sets And Applications: Selected Papers by Lotfi A. Zadeh of [Yager et al., 1987]. pages in book: 105-146.

[Zadeh, 1975a] Zadeh, L. A. (1975a). *The Concept of a Linguistic Variable and its Application to Approximate Reasoning (Part 1)*, pages 199–249. Volume Fuzzy Sets And Applications: Selected Papers by Lotfi A. Zadeh of [Yager et al., 1987]. pages in book: 219-270.

[Zadeh, 1975b] Zadeh, L. A. (1975b). *The Concept of a Linguistic Variable and its Application to Approximate Reasoning (Part 2)*, pages 301–357. Volume Fuzzy Sets And Applications: Selected Papers by Lotfi A. Zadeh of [Yager et al., 1987]. pages in book: 271-327.

[Zadeh, 1975c] Zadeh, L. A. (1975c). *The Concept of a Linguistic Variable and its Application to Approximate Reasoning (Part 3)*, pages 43–80. Volume Fuzzy Sets And Applications: Selected Papers by Lotfi A. Zadeh of [Yager et al., 1987]. pages in book: 329-366.

[Zadeh, 1977] Zadeh, L. A. (1977). Similarity relations and fuzzy orderings. In [Yager et al., 1987], pages 177–200. pages in book: 81-104.

[Zadeh, 1979] Zadeh, L. A. (1979). A theory of approximate reasoning. In [Yager et al., 1987], pages 149–194. pages in book: 367-412.